aws

ユーザーガイド

Amazon Simple Storage Service

API バージョン 2006-03-01

Amazon Simple Storage Service: ユーザーガイド

Table of Contents

Amazon S3 とは

Amazon Simple Storage Service (Amazon S3) は、業界をリードするスケーラビリティ、データ可用性、セキュリティ、およびパフォーマンスを提供するオブジェクトストレージサービスです。あらゆる規模や業界のお客様が、Amazon S3 を使用して、データレイク、ウェブサイト、モバイルアプリケーション、バックアップおよび復元、アーカイブ、エンタープライズアプリケーション、IoT デバイス、ビッグデータ分析など、広範なユースケースのデータを容量にかかわらず、保存して保護することができます。Amazon S3 には、特定のビジネス、組織、コンプライアンスの要件を満たすために、データへのアクセスを最適化、整理、設定できる管理機能があります。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

トピック

Amazon S3 の機能

ストレージクラス

Amazon S3 では、さまざまなユースケース向けに、幅広いストレージクラスが提供されています。例えば、ミッションクリティカルな本番環境のデータを S3 Standard または S3 Express One Zone に保存して頻繁にアクセスしたり、アクセス頻度の低いデータを S3 標準 – IA または S3 One Zone-

IA に保存してコストを節約したり、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive に最も低いコストでデータをアーカイブしたりできます。

Amazon S3 Express One Zone は、最もレイテンシーの影響を受けやすいアプリケーションに 1 桁のミリ秒単位で一貫したデータアクセスを提供することを目的として構築された、高パフォーマンスのシングルアベイラビリティーゾーンの Amazon S3 ストレージ クラスです。S3 Express One Zone は、現在入手可能なレイテンシーが最も低いクラウドオブジェクトストレージクラスで、データアクセス速度は最大 10 倍速く、リクエストコストは S3 スタンダードよりも 50% 低減されます。S3 Express One Zone は、オブジェクトストレージをコンピュートリソースと同じ場所に配置するオプションを備えた単一のアベイラビリティーゾーンを選択できる最初の S3 ストレージクラスです。これにより、最高レベルのアクセス速度を実現できます。アクセス速度をさらに向上させ、1 秒あたり数十万ものリクエストをサポートするために、データは新しいバケットタイプ、つまり Amazon S3 ディレクトリバケットに保存されます。詳細については、S3 Express One Zone とはおよびディレクトリバケットを参照してください。

S3 Intelligent-Tiering では、変更する、または不明なアクセスパターンを持つデータを保存できます。これにより、アクセスパターンが変更されたときに 4 つのアクセス層間でデータを自動的に移動することで、ストレージコストを最適化できます。4 つのアクセス階層にある、高頻度のアクセスと低頻度のアクセス用に最適化された 2 つの低レイテンシーのアクセス階層と、稀にしかアクセスされないデータ向けに設計された非同期アクセス用の 2 つのオプトインアーカイブアクセス階層があります。

詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。

ストレージ管理

Amazon S3 には、コストの管理、規制要件への対応、レイテンシーの削減、コンプライアンス要件のためのデータの複数の個別コピーの保存で使用できるストレージ管理機能があります。

- S3 ライフサイクル – オブジェクトを管理し、ライフサイクルを通じてコスト効率の高い方法で保存できるようにライフサイクル設定を使用します。オブジェクトを他の S3 ストレージクラスに移行したり、ライフタイムが終了したオブジェクトを期限切れにすることができます。
- S3 オブジェクトロック – Amazon S3 オブジェクトが固定期間または無期限に削除または上書きされるのを防止します。オブジェクトロックを使用して、write-once-read-many (WORM) ストレージを必要とする規制要件を満たしたり、オブジェクトの変更や削除に対する保護レイヤーを追加したりできます。
- S3 レプリケーション – オブジェクトおよびそれぞれのメタデータタグとオブジェクトタグを、同じまたは異なる AWS リージョン にある 1 つまたは複数のレプリケーション先バケットにレプリ

ケートして、レイテンシーの削減、コンプライアンス、セキュリティ、その他のユースケースで活用できます。

- S3 バッチ操作 – 1 つの S3 API リクエストまたは Amazon S3 コンソールで数回クリックするだけで、数十億のオブジェクトを大規模に管理できます。バッチ操作を使用すると、次のようなオペレーションを実行できます。コピー、AWS Lambda 関数の呼び出し、数百万または数十億のオブジェクトの復元。

アクセス管理とセキュリティ

Amazon S3 には、バケットとオブジェクトへのアクセスを監査および管理する機能があります。デフォルトでは、S3 バケットとオブジェクトはプライベートです。作成した S3 リソースにのみアクセスできます。以下の機能を使用して、特定のユースケースをサポートする詳細なリソース許可を付与したり、Amazon S3 リソースの許可を監査したりできます。

- S3 ブロックパブリックアクセス – S3 バケットおよびオブジェクトへのパブリックアクセスをブロックします。デフォルトでは、[パブリックアクセスをブロック] 設定はバケットレベルで有効になっています。ユースケースでオフにする必要のある設定が 1 つ以上あることがわかっている場合を除き、すべての [パブリックアクセスをブロック] 設定を有効にしておくことをお勧めします。詳細については、「S3 バケットへのパブリックアクセスブロック設定の構成」を参照してください。
- AWS Identity and Access Management (IAM) — IAM は、AWS リソース (Amazon S3 リソースなど) へのアクセスを安全に管理するためのウェブサービスです。IAM を使用すると、ユーザーがアクセスできる AWS のリソースを制御するアクセス許可を集中管理できます。IAM を使用して、誰を認証 (サインイン) し、誰にリソースの使用を認可する (アクセス許可を付与する) かを制御します。
- バケットポリシー – IAM ベースのポリシー言語を使用して、S3 バケットとその中のオブジェクトに対するリソースベースの許可を設定します。
- Amazon S3 アクセスポイント — 専用アクセスポリシーを持つ名前付きネットワークエンドポイントを設定して、Amazon S3 の共有データセットへの大規模なデータアクセスを管理します。
- アクセスコントロールリスト (ACL) – 個々のバケットおよびオブジェクトに対する読み取りおよび書き込みの許可を、承認されたユーザーに付与します。原則として、アクセスコントロールには ACL ではなく S3 リソースベースのポリシー (バケットポリシーとアクセスポイントポリシー) または IAM ユーザーポリシーを使用することをお勧めします。ポリシーとは、よりシンプルで柔軟なアクセス制御のオプションです。バケットポリシーとアクセスポイントポリシーを使用すると、Amazon S3 リソースに対するすべてのリクエストに広く適用されるルールを定義できます。

リソースベースのポリシーまたは IAM ユーザーポリシーの代わりに ACL を使用する場合の特定のケースの詳細については、「ACL によるアクセス管理」を参照してください。

- S3 オブジェクト所有権 - バケット内のすべてのオブジェクトの所有権を取得し、Amazon S3 に保存されているデータのアクセス管理を簡素化します。S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、ACL を無効または有効にするのに使用できます。デフォルトでは、ACL は無効になっています。ACL を無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。
- IAM Access Analyzer for S3 – S3 バケットアクセスポリシーを評価およびモニタリングし、ポリシーが S3 リソースへの意図したアクセスのみを提供することを確認します。

データ処理

データを変換し、ワークフローをトリガーして、他のさまざまな処理アクティビティを大規模に自動化するには、次の機能を使用できます。

- S3 Object Lambda – S3 GET、HEAD、LIST リクエストに独自のコードを追加して、データがアプリケーションに返されるときにそのデータを変更および処理できます。行のフィルタリング、画像の動的なサイズ変更、機密データの編集などを行います。
- イベント通知 – S3 リソースに変更が加えられると、Amazon Simple Notification Service (Amazon SNS)、Amazon Simple Queue Service (Amazon SQS)、および AWS Lambda を使用するワークフローをトリガーします。

ストレージのログ記録とモニタリング

Amazon S3 には、Amazon S3 リソースの使用状況をモニタリングおよびコントロールするために使用できるロギングおよびモニタリングツールが用意されています。詳細については、「モニタリングツール 」を参照してください。

自動モニタリングツール

- Amazon S3 の Amazon CloudWatch メトリクス – TS3 リソースのオペレーション状態を追跡し、推定請求額がユーザー定義のしきい値に達したときに請求アラートを設定します。
- AWS CloudTrail – ユーザー、ロール、またはAmazon S3 で AWS のサービス によって行われるアクションを記録します。CloudTrail ログを使用すると、S3 バケットレベルおよびオブジェクトレベルのオペレーションの詳細な API 追跡が可能になります。

手動モニタリングツール

- サーバーアクセスログ – バケットに対するリクエストの詳細が記録されます。サーバーアクセスのログ記録を使用して、セキュリティとアクセスの監査、カスタマーベースに関するラーニング、Amazon S3 請求書の把握などの多くのユースケースに対応できます。
- AWSTrusted Advisor – AWS ベストプラクティスチェックを使用してアカウントを評価し、AWS インフラストラクチャを最適化し、セキュリティとパフォーマンスを向上させ、コストを削減し、サービスクォータを監視する方法を特定します。その後、推奨事項に従って、サービスとリソースを最適化できます。

分析とインサイト

Amazon S3 には、ストレージの使用状況を可視化するための機能が用意されています。これにより、ストレージを大規模に理解、分析し、最適化することができます。

- Amazon S3 Storage Lens – ストレージを理解、分析し、最適化します。S3 ストレージレンズは、使用状況およびアクティビティに関する 60 以上のメトリクスとインタラクティブなダッシュボードを提供し、組織全体、特定のアカウント、AWS リージョン、バケット、またはプレフィックスに関するデータを集約します。
- ストレージクラス分析 – ストレージアクセスパターンを分析して、よりコスト効果の高いストレージクラスにデータを移動するタイミングを決定します。
- インベントリ付き S3 インベントリレポート – オブジェクトとそれに対応するメタデータを監査してレポートし、インベントリレポートでアクションを実行するように他の Amazon S3 機能を設定します。例えば、オブジェクトのレプリケーションと暗号化のステータスをレポートできます。インベントリレポートの各オブジェクトで使用できるすべてのメタデータのリストについては、「Amazon S3 インベントリリスト」を参照してください。

強力な整合性

Amazon S3 には、すべての AWS リージョン にある Amazon S3 バケットの、オブジェクトの PUT と DELETE に関する、書き込み後読み取りの強力な整合性があります。この動作は、新しいオブジェクトへの書き込みと、既存のオブジェクトを上書きする PUT、そして DELETE リクエストにも適用されます。さらに、Amazon S3 Select、Amazon S3 アクセスコントロールリスト (ACL)、Amazon S3 オブジェクトタグ、オブジェクトメタデータ (HEAD オブジェクトなど) での読み込みオペレーションには、強力な整合性があります。詳細については、「Amazon S3 のデータ整合性モデル」を参照してください。

Amazon S3 の仕組み

Amazon S3 は、データをオブジェクトとしてバケットに保存するオブジェクトストレージサービスです。オブジェクトとは、ファイルと、そのファイルを記述している任意のメタデータのことです。バケット とは、オブジェクトのコンテナのことです。

Amazon S3 にデータを保存するには、まずバケットを作成し、バケット名および AWS リージョンを指定します。次に、Amazon S3 のオブジェクトとしてそのバケットにデータをアップロードします。各オブジェクトには、キー(またはキー名) があります。これは、バケット内のオブジェクトの一意の識別子です。

S3 には、特定のユースケースをサポートするように設定できる機能があります。例えば、S3 Versioning を使用すると、オブジェクトの複数のバージョンを同じバケットに保持し、誤って削除または上書きされたオブジェクトを復元することができます。

バケットとその中のオブジェクトはプライベートであり、アクセス許可を明示的に付与した場合にのみアクセスできます。バケットポリシー、AWS Identity and Access Management(IAM) ポリシー、アクセスコントロールリスト (ACL)、および S3 アクセスポイントを使用して、アクセスを管理できます。

トピック

バケット

バケットとは、Amazon S3 に保存されるオブジェクトのコンテナです。バケットにはオブジェクトをいくつでも保存でき、アカウントにはバケットを 100 個まで保存できます。増加をリクエストするには、Service Quotas コンソールにアクセスしてください。

すべてのオブジェクトはバケット内に保存されます。例えば、photos/puppy.jpg という名前のオブジェクトが米国西部 (オレゴン) リージョンにある DOC-EXAMPLE-BUCKET バケットに保存される場合、URL https://DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com/photos/puppy.jpg を使用してアドレスを解決できます。詳細については、「バケットへのアクセス」を参照してください。

バケットを作成するときは、バケット名を入力し、バケットが存在する AWS リージョン を選択します。一度バケットを作成したら、そのバケット名またはリージョンを変更することはできません。バケット名は、バケットの命名規則に従う必要があります。また、バケットを設定して、S3 バージョニングまたは、他のストレージ管理機能を使用できます。

バケットは、以下も行います。

- 最も高いレベルで Amazon S3 名前空間を編成します。
- ストレージおよびデータ転送料金が課金されるアカウントを特定します。
- バケットポリシー、アクセスコントロールリスト (ACL)、および S3 アクセスポイントなどのアクセスコントロールオプションを提供します。これを使用すると、Amazon S3 リソースへのアクセスを管理できます。
- 使用状況レポートの集計単位として機能します。

バケットの詳細については、「バケットの概要」を参照してください。

オブジェクト

オブジェクトとは、Amazon S3 に保存される基本エンティティです。オブジェクトは、オブジェクトデータとメタデータで構成されます。メタデータは、オブジェクトを表現する名前と値のペアのセットです。これには最終更新日などのデフォルトメタデータや、Content-Type などの標準 HTTP メタデータが含まれます。また、オブジェクトの保存時にカスタムメタデータを指定することもできます。

オブジェクトは、バケット内でキー (名前)とバージョン ID (バケットで S3 バージョニングが有効になっている場合) によって一意に特定されます。オブジェクトの詳細については、Amazon S3 オブジェクトの概要 を参照してください。

キー

オブジェクトキー (または キー名) は、バケット内のオブジェクトの固有の識別子です。バケット内のすべてのオブジェクトは、厳密に 1 個のキーを持ちます。バケット、オブジェクトキー、および

オプションでバージョン ID (バケットで S3 バージョニングが有効になっている場合) の組み合わせによって、各オブジェクトが一意に識別されます。そのため、Amazon S3 を「バケット + キー + バージョン」とオブジェクト自体の間での基本データマップと考えることができます。

Amazon S3 内の各オブジェクトは、ウェブサービスエンドポイント、バケット名、キー、およびオプションでバージョンを組み合わせることで一意にアドレスを指定できます。例えば、`https://DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com/photos/puppy.jpg` という URL で、「`DOC-EXAMPLE-BUCKET`」がバケットの名前で、「`photos/puppy.jpg`」がキーです。

オブジェクトキーの詳細については、「オブジェクトキー名の作成」を参照してください。

S3 バージョニング

同じバケット内でオブジェクトの複数のバリアントを保持するには、S3 バージョニングを使用します。S3 バージョニングを使用すると、 バケットに保存されたあらゆるオブジェクトのあらゆるバージョンを保存、取得、復元することができます。バージョニングを使用すれば、意図しないユーザーアクションからもアプリケーション障害からも、簡単に復旧できます。

詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

バージョン ID

バケットで S3 バージョニングを有効にすると、Amazon S3 はバケットに追加されたすべてのオブジェクトに一意のバージョン ID を与えます。バージョニングを有効にした時点でバケットにすでに存在していたオブジェクトのバージョン ID は `null` です。これらの (またはその他の) オブジェクトを他のオペレーション (CopyObject および PutObject) で変更すると、新しいオブジェクトは一意のバージョン ID を取得します。

詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

バケットポリシー

バケットポリシーは、リソースベースの AWS Identity and Access Management (IAM) ポリシーを使用して、バケットとその中のオブジェクトへのアクセス許可を付与できます。バケット所有者のみが、ポリシーをバケットに関連付けることができます。バケットに添付された許可は、バケット所有者が所有するバケットのすべてのオブジェクトに適用されます。バケットポリシーのサイズは 20 KB に制限されています。

バケットポリシーは、AWS で標準である JSON ベースのアクセスポリシー言語を使用しています。バケットポリシーを使用して、バケット内のオブジェクトに対する許可を追加または拒否できます。

バケットポリシーは、リクエスタ、S3 アクション、リソース、リクエストの側面または条件(リクエストの作成に使用された IP アドレスなど)など、ポリシー内のエレメントに基づいてリクエストを許可または拒否します。例えば、バケット所有者がアップロードされたオブジェクトを完全にコントロールできるように、S3 バケットにオブジェクトをアップロードするクロスアカウント許可を付与するバケットポリシーを作成できます。詳細については、Amazon S3 バケットポリシーの例 を参照してください。

バケットポリシーでは、Amazon リソースネーム (ARN) やその他の値に対してワイルドカード文字を使用して、オブジェクトのサブセットに対する許可を付与できます。例えば、共通のプレフィックスで始まるか、.html などの特定の拡張子で終わるオブジェクトのグループへのアクセスをコントロールできます。

S3 アクセスポイント

Amazon S3 アクセスポイントは、そのエンドポイントを使用してデータにアクセスする方法を説明する専用のアクセスポリシーを持つ名前付きネットワークエンドポイントです。アクセスポイントは、バケットにアタッチされ、それを使用して、GetObject や PutObject などの S3 オブジェクト操作を実行できます。アクセスポイントは、Amazon S3 の共有データセットへの大規模なデータアクセスの管理を簡素化します。

各アクセスポイントには独自のアクセスポイントポリシーがあります。また、アクセスポイントごとにブロックパブリックアクセス設定を設定することもできます。仮想プライベートクラウド (VPC) からのリクエストだけを受け入れるようにアクセスポイントを設定することで、プライベートネットワークへの Amazon S3 データアクセスを制限できます。

詳細については、「Amazon S3 アクセスポイントを使用したデータアクセスの管理」を参照してください。

アクセスコントロールリスト (ACL)

ACL を使用して、個々のバケットとオブジェクトの読み取りと書き込みの許可を認可されたユーザーに付与できます。各バケットとオブジェクトには、サブリソースとして ACL がアタッチされています。ACL は、アクセスを付与する AWS アカウント またはグループまたはアクセスのタイプを定義します。ACL は IAM よりも優先されるアクセスコントロールメカニズムです。ACL の詳細については、「アクセスコントロールリスト (ACL) の概要」を参照してください。

S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、バケットにアップロードされる新しいオブジェクト所有権を制御し、ACL を無効にするのに使用できます。デフォルトでは、オブジェクト所有権はバケット所有者の強制設定に設定され、すべての ACL は無効になります。ACL を

無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。

Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。ACL を無効にすると、誰がオブジェクトをバケットにアップロードしたかに関係なく、ポリシーを使用してバケット内のすべてのオブジェクトへのアクセスを制御できます。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

リージョン

作成したバケットを Amazon S3 が保存する地理的な AWS リージョン を選択できます。レイテンシーを最適化し、コストを最小限に抑えて規制用件に対応できるリージョンを選ぶとよいでしょう。明示的に別のリージョンに移動またはレプリケートする場合を除き、AWS リージョン に保存されたオブジェクトは、そのリージョンから移動されることはありません。たとえば、欧州 (アイルランド) リージョンに格納されたオブジェクトは、ずっとそのリージョンに置かれたままです。

> **Note**
>
> 自分のアカウントで有効になっている AWS リージョン では、Amazon S3 とその機能にのみアクセスできます。リージョンで AWS リソースを作成および管理できるようにする方法の詳細については、「AWS 全般のリファレンス」の「AWS リージョン の管理」を参照してください。

Amazon S3 のリージョンとエンドポイントのリストについては、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。

Amazon S3 のデータ整合性モデル

Amazon S3 には、すべての AWS リージョン にある Amazon S3 バケットの、オブジェクトの PUT と DELETE に関する、書き込み後読み取りの強力な整合性があります。この動作は、新しいオブジェクトへの書き込みと、既存のオブジェクトを上書きする PUT リクエスト、そして DELETE リクエストにも適用されます。さらに、Amazon S3 Select、Amazon S3 アクセスコントロールリスト (ACL)、Amazon S3 オブジェクトタグ、オブジェクトメタデータ (HEAD オブジェクトなど) での読み込みオペレーションには、強力な整合性があります。

単一のキーに対する更新はアトミックです。例えば、あるスレッドから既存のキーに PUT リクエストを実行し、同時に同じキーに対して別のスレッドから GET リクエストを実行すると、古いデータ

または新しいデータを取得できますが、データの一部分だけが取得されることも、破損することもありません。

Amazon S3 は、AWS データセンターに配置された複数のサーバー間でデータを複製することにより、高可用性を実現します。PUT リクエストが成功した場合、データは安全に保存されています。成功した PUT 応答の受信後に開始された読み取り (GET または LIST) は、PUT リクエストによって書き込まれたデータを返します。この動作の例を示します。

- 新しいオブジェクトを Amazon S3 に書き込み、すぐにバケット内のキーを一覧表示します。新しいオブジェクトがリストに表示されます。
- 既存のオブジェクトを置換し、すぐにそのオブジェクトの読み取りを試みます。Amazon S3 が新しいデータを返します。
- 既存のオブジェクトを削除し、すぐにそのオブジェクトの読み取りを試みます。オブジェクトが削除されたため、Amazon S3 はデータを返しません。
- 既存のオブジェクトを削除し、すぐにバケット内のキーのリストを表示します。オブジェクトはリストに表示されません。

> **Note**
>
> - Amazon S3 は、同時書き込みのオブジェクトロックをサポートしていません。同じキーに対して 2 つの PUT リクエストが同時に行われた場合、最新のタイムスタンプを持つリクエストが優先されます。これが問題になる場合は、アプリケーション内にオブジェクトロックメカニズムを構築する必要があります。
> - 更新はキーベースです。複数キーにまたがるアトミックな更新を行う方法はありません。たとえば、ご自分で機能をアプリケーション設計に組み込まない限り、別のキーの更新に依存してキーを更新することはできません。

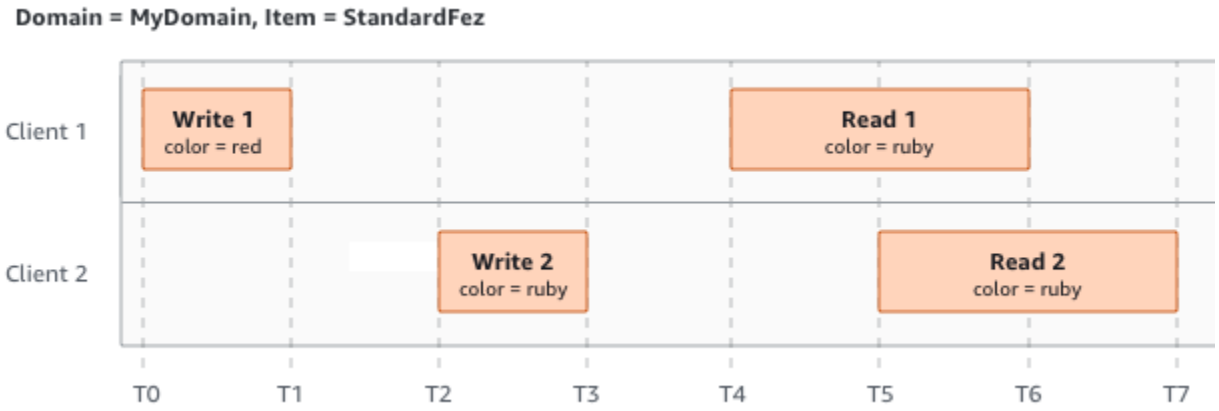
バケット設定には、結果整合性モデルがあります。具体的には、次のように処理されます。

- バケットを削除してすぐにすべてのバケットを一覧表示しても、削除されたバケットは引き続きリストに表示されます。
- バケットで初めてバージョニングを有効にしたときは、変更が完全に反映されるまでに、少し時間がかかることがあります。バケットへのオブジェクトの書き込みオペレーション (PUT または DELETE リクエスト) は、バージョニングを有効にして 15 分待ってから発行することをお勧めします。

アプリケーションの同時実行

このセクションでは、同じアイテムに複数のクライアントから書き込むときに、Amazon S3 で予想される動作の例を示します。

次の例では、R1 (読み取り 1) と R2 (読み取り 2) の開始前に W1 (書き込み 1) と W2 (書き込み 2) が完了しています。S3 には強固な整合性があるため、R1 と R2 はどちらも `color = ruby` を返します。



次の例では、R1 の開始前に、W2 は完了していません。したがって、R1 は `color = ruby` または `color = garnet` を返す可能性があります。ただし、R2 が開始する前に W1 と W2 が終了するため、R2 は `color = garnet` を返します。

最後の例では、W1 が受信確認を受け取る前に、W2 が開始します。したがって、これらは同時の書き込みとみなされます。どの書き込みを優先するのかを決定するにあたり、Amazon S3 は内部的に last-writer-wins セマンティクスを使用します。しかし、ネットワークレイテンシーなどのさまざまな要因により、Amazon S3 がリクエストを受信する順序や、アプリケーションが受信確認を受け取る順序を予測することはできません。例えば、W2 が同じリージョンにある Amazon EC2 インスタン

スによって開始される一方で、W1 は遠くにあるホストによって開始されるかもしれません。両方の書き込みの受信確認を受け取った後に、読み込みを実行することが、最終的な値を決定する最善の方法です。

関連サービス

Amazon S3 にロードしたデータは、他の AWS のサービスでも利用できます。よく使用すると思われるサービスは次のとおりです。

- Amazon Elastic Compute Cloud (Amazon EC2) – 安全でスケーラブルなコンピューティング容量を AWS クラウド で提供します。Amazon EC2 の使用により、ハードウェアに事前投資する必要がなくなり、アプリケーションをより速く開発およびデプロイできます。Amazon EC2 を使用すると、必要な数 (またはそれ以下) の仮想サーバーの起動、セキュリティおよびネットワーキングの構成、ストレージの管理ができます。
- Amazon EMR – ビジネス、研究者、データアナリスト、およびデベロッパーが、簡単かつ費用対効果の高い方法で、莫大な量のデータを処理できます。Amazon EMR は、Amazon EC2 および Amazon S3 のウェブスケールのインフラストラクチャ上で稼動するホストされた Hadoop フレームワークを使用しています。
- AWSSnow ファミリー– 厳しいデータセンター以外の環境や、一貫性のあるネットワーク接続がない場所で運用を実行する必要があるお客様を支援します。AWS Snow ファミリーデバイスを使用して、インターネットに接続できない環境で、ローカルでコスト効果の高い方法で AWS クラウドのストレージおよび処理能力にアクセスできます。
- AWS Transfer Family – セキュアシェル (SSH) ファイル転送プロトコル (SFTP)、SSL 経由ファイル転送プロトコル (FTPS)、およびファイル転送プロトコル (FTP) を使用して、Amazon S3 または Amazon Elastic File System (Amazon EFS) との間で直接ファイル転送を行う完全マネージドサポートを提供します。

Amazon S3 へのアクセス

Amazon S3 は次のいずれかの方法で使用できます。

AWS Management Console

コンソールは、Amazon S3 と AWS リソースを管理するためのウェブベースのユーザーインターフェイスです。AWS アカウント にサインアップ済みの場合は、AWS Management Console にサインインし、AWS Management Console ホームページから [S3] を選択することで、Amazon S3 コンソールにアクセスできます。

AWS Command Line Interface

AWS コマンドラインツールを使用して、コマンドを発行するか、システムのコマンドラインでスクリプトを作成して AWS (S3 を含む) タスクを実行します。

AWS Command Line Interface (AWS CLI) は、幅広い AWS のサービス のセットに対するコマンドを提供します。AWS CLI は、Windows、macOS、Linux でサポートされています。使用を開始するには、「 AWS Command Line Interfaceユーザーガイド」を参照してください。Amazon S3 用コマンドの詳細については、AWS CLIコマンドリファレンスの s3api および s3control を参照してください。

AWS SDK

AWS には、さまざまなプログラミング言語およびプラットフォーム (Java、Python、Ruby、.NET、iOS、Android など) のライブラリとサンプルコードで構成された SDK (ソフトウェア開発キット) が用意されています。AWS SDK は、S3 や AWS へのプログラムによるアクセスを作成するのに役立ちます。Amazon S3 は REST サービスです。AWS SDK ライブラリを使用して Amazon S3 にリクエストを送信できます。これは、基盤となる Amazon S3 REST API をラップし、プログラミングタスクを簡素化します。例えば、SDK は署名の計算、リクエストの暗号化による署名、エラーの管理、リクエストの自動再試行などのタスクを処理します。AWS SDK のダウンロードやインストールなどの詳細については、「AWS のツール」を参照してください。

Amazon S3 とのすべてのやり取りは認証されるか匿名で行われます。AWS SDK を使用している場合、指定したキーから、ライブラリによって認証のための署名が計算されます。Amazon S3 へのリクエストの作成方法の詳細については、「リクエストの実行」を参照してください。

Amazon S3 REST API

Amazon S3 は、プログラミング言語に依存しないアーキテクチャとして設計されており、AWS がサポートされているインターフェイスを使用してオブジェクトを保存、取得します。Amazon S3 REST API を使用して、プログラムによって S3 や AWS にアクセスすることができます。REST API は、Amazon S3 に対する HTTP インターフェイスです。REST API では、標準 HTTP リクエストを使用してバケットとオブジェクトを作成、取得、削除できます。

REST API を使用する場合、HTTP をサポートする任意のツールキットを使用できます。匿名で読み取り可能なオブジェクトであれば、ブラウザを使用して取得することもできます。

REST API は標準の HTTP ヘッダーとステータスコードを使用するため、標準のブラウザとツールキットが予期したとおりに機能します。一部のエリアでは、HTTP に機能が追加されています (たとえば、アクセスコントロールをサポートするヘッダーを追加しました)。このように新機能を追加する場合、できるだけ標準 HTTP 書式の使用法に合致するように最善を尽くしました。

ただし、アプリケーションで直接 REST API を呼び出す場合、署名を計算するコードを作成し、それをリクエストに追加する必要があります。Amazon S3 へのリクエストの作成方法の詳細については、「リクエストの実行」を参照してください。

> **Note**
>
> SOAP API のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。新しい Amazon S3 機能は、SOAP ではサポートされません。REST API か AWS SDK を使用することをお勧めします。

Amazon S3 の支払い

Amazon S3 の料金は、アプリケーションのストレージ要件を考慮しなくてすむように設定されています。ほとんどのストレージプロバイダーでは、あらかじめ決められた量のストレージおよびネットワーク転送容量を購入する必要があります。このシナリオでは、その容量を超えると、サービスが停止されるか、高額な超過料金を支払う必要があります。その容量を超えない場合でも、全量を使用したものとして支払うことになります。

Amazon S3 では、実際に使用した分だけが請求されます。隠れた料金や超過料金はありません。このモデルでは、AWS インフラストラクチャのコスト面のメリットを得ながら、ビジネスの成長に応じた可変コストのサービスを利用することができます。詳細については、Amazon S3 の料金 を参照してください。

AWS にサインアップすると、Amazon S3 を含む AWS のすべてのサービスに対して AWS アカウント が自動的にサインアップされます。ただし、料金が発生するのは実際に使用したサービスの分だけです。Amazon S3 の新規のお客様は、Amazon S3 を無料で使い始めることができます。詳細については、「AWS 無料利用枠」を参照してください。

請求を表示するには、AWS Billing and Cost Management コンソールで請求およびコスト管理ダッシュボードに移動します。AWS アカウント 請求の詳細については、AWS Billing ユーザーガイドを参照してください。AWS 請求および AWS アカウント についてご質問がある場合は、AWS Support にお問い合わせください。

PCI DSS コンプライアンス

Amazon S3 は、マーチャントまたはサービスプロバイダーによるクレジットカードデータの処理、ストレージ、および伝送をサポートしており、Payment Card Industry (PCI) Data Security Standard (DSS) に準拠していることが確認されています。PCI DSS の詳細 (AWS PCI Compliance Package のコピーをリクエストする方法など) については、「PCI DSS レベル 1」を参照してください。

Amazon S3 の開始方法

Amazon S3 は、バケットとオブジェクトを操作することで、使用を開始できます。バケット とは、オブジェクトのコンテナのことです。オブジェクト とは、ファイルと、そのファイルを記述している任意のメタデータのことです。

Amazon S3 にオブジェクトを保存するには、バケットを作成し、そのバケットにオブジェクトをアップロードします。オブジェクトがバケットの中にあるときは、オブジェクトを開き、ダウンロードして、移動させます。オブジェクトまたはバケットが不要になったら、リソースをクリーンアップします。

Amazon S3 では、お支払いは実際に使用した分のみです。Amazon S3 の機能と料金の詳細については、「Amazon S3」を参照してください。Amazon S3 の新規のお客様は、Amazon S3 を無料で使い始めることができます。詳細については、AWS 無料利用枠 を参照してください。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

動画: Amazon S3 の開始方法

前提条件

開始する前に、必ず「前提条件: Amazon S3 をセットアップする」の手順を完了させてください。

トピック

前提条件: Amazon S3 をセットアップする

AWS にサインアップすると、Amazon S3 を含む AWS のすべてのサービスに対して AWS アカウント が自動的にサインアップされます。料金が発生するのは、実際に使用したサービスの分のみです。

Amazon S3 では、お支払いは実際に使用した分のみです。Amazon S3 の機能と料金の詳細については、Amazon S3 を参照してください。Amazon S3 の新規のお客様は、Amazon S3 を無料で使い始めることができます。詳細については、「AWS 無料利用枠」を参照してください。

Amazon S3 をセットアップするには、以下のセクションのステップを使用します。

AWS にサインアップして Amazon S3 をセットアップする際に、オプションで AWS Management Console の表示言語を変更できます。詳細については、AWS Management Console 開始方法のガイドの AWS Management Console の言語の変更を参照してください。

トピック

AWS アカウントへのサインアップ

AWS アカウントがない場合は、以下のステップを実行して作成します。

AWS アカウントにサインアップするには

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

   AWS アカウントにサインアップすると、AWS アカウントのルートユーザーが作成されます。ルートユーザーには、アカウントのすべてのAWS のサービスとリソースへのアクセス権があります。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行してください。

サインアップ処理が完了すると、AWS からユーザーに確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] を選んで、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理できます。

管理アクセス権を持つユーザーを作成する

AWS アカウント にサインアップしたら、AWS アカウントのルートユーザー をセキュリティで保護し、AWS IAM Identity Center を有効にして、管理ユーザーを作成します。これにより、日常的なタスクにルートユーザーを使用しないようにします。

AWS アカウントのルートユーザーをセキュリティで保護する

1. [ルートユーザー] を選択し、AWS アカウントのメールアドレスを入力して、アカウント所有者として AWS Management Console にサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「ルートユーザーとしてサインインする」を参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、IAM ユーザーガイドの「AWS アカウント のルートユーザーの仮想 MFA デバイスを有効にする (コンソール)」を参照してください。

管理アクセス権を持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Center の有効化」を参照してください。
2. IAM アイデンティティセンターで、ユーザーに管理アクセス権を付与します。

   IAM アイデンティティセンターディレクトリ をアイデンティティソースとして使用するチュートリアルについては、「AWS IAM Identity Center ユーザーガイド」の「デフォルト IAM アイデンティティセンターディレクトリを使用したユーザーアクセスの設定」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「AWS アクセスポータルにサインインする」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するベストプラクティスに従うアクセス許可セットを作成します。

   「AWS IAM Identity Center ユーザーガイド」の「Create a permission set」の手順に従ってください。

2. ユーザーをグループに割り当ててから、そのグループにシングルサインオンアクセスを割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「Add groups」を参照してください。

ステップ 1: 最初の S3 バケットを作成する

AWS にサインアップしたら、AWS Management Console を使っていつでも Amazon S3 にバケットを作成できます。Amazon S3 のオブジェクトはすべてバケットに保管されます。Amazon S3 にデータを保管する前に、バケットを作成する必要があります。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

> **Note**
>
> バケットの作成は課金対象にはなりません。バケット内にオブジェクトを保存した場合、およびバケット宛てまたはバケットからオブジェクトを転送した場合にのみ課金されます。このガイドの例に従って操作して発生する使用料はごくわずかです (1 USD 未満)。ストレージ料金の詳細については、「Amazon S3 の料金」を参照してください。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ページ上部にあるナビゲーションバーで、現在表示されている AWS リージョン の名前をクリックします。次に、バケットを作成するリージョンを選択します。

   > **Note**
   >
   > レイテンシーとコストを最小化するため、さらに規制条件に対応するために、最寄りのリージョンを選択します。明示的に別のリージョンに移動する場合を除き、特定のリージョンに保管されたオブジェクトは、そのリージョンから移動されることはありません。Amazon S3 AWS リージョン のリストについては、Amazon Web Services 全般のリファレンス の「AWS のサービス エンドポイント」を参照してください。

3. 左側のナビゲーションペインで、[バケット] を選択します。
4. [Create bucket (バケットの作成)] を選択します。

   [バケットを作成する] ページが開きます。
5. [全般設定] で、バケットが作成される AWS リージョン を確認します。
6. [バケットタイプ] で、[汎用] を選択します。
7. [バケット名] にバケットの名前を入力します。

   バケット名には次の条件があります。

   - パーティション内で一意にする必要があります。パーティションは、リージョンのグループです。AWS には、現在、aws (標準リージョン)、aws-cn (中国リージョン)、および aws-us-gov (AWS GovCloud (US) Regions) の 3 つのパーティションがあります。
   - 3 ~ 63 文字で指定する。
   - 小文字、数字、ドット (.)、およびハイフン (-) のみで構成できます。互換性を最も高くするには、静的ウェブサイトホスティング専用のバケットを除き、バケット名にドット (.) を使用しないことをお勧めします。
   - 文字や数字で始まり、文字や数字で終わります。

   バケットを作成したら、その名前を変更することはできません。バケットの命名の詳細については、「バケットの名前付け」を参照してください。

> **Important**
>
> バケット名にアカウント番号などの機密情報を含めないでください。バケット名は、バケット内のオブジェクトを参照する URL に表示されます。

8. AWS Management Console では、既存のバケットの設定を新しいバケットにコピーできます。既存のバケットの設定をコピーしない場合は、次のステップにスキップします。

> **Note**
>
> このオプションの特徴:
>
> - AWS CLI では使用できません。コンソールでのみ利用できます。
> - ディレクトリバケットには利用できません。
> - バケットポリシーは既存のバケットから新しいバケットにコピーしません。

既存のバケットの設定をコピーするには、[既存のバケットから設定をコピー] で [バケットを選択] をクリックします。[バケットを選択] ウィンドウが開きます。コピーする設定を持つバケットを検索して、[バケットを選択] をクリックします。[バケットを選択] ウィンドウが閉じて、[バケットを作成] ウィンドウが再び開きます。

[既存のバケットから設定をコピー] に、選択したバケットの名前が表示されるようになります。コピーしたバケット設定を削除するための [デフォルトを復元] オプションも表示されます。[バケットを作成] ページで、バケットの残りの設定を確認します。ここで、選択したバケットの設定と一致していることを確認できます。最後のステップにスキップできます。

9. オブジェクト所有者 で、ACL を無効または有効にし、バケットにアップロードされたオブジェクトの所有権を制御するには、次のいずれかの設定を選択します。

ACL は無効です

- バケット所有者強制 (デフォルト) – ACL は無効になり、バケット所有者はバケット内のすべてのオブジェクトを自動的に所有し、完全に制御します。ACL は、S3 バケット内のデータに対するアクセス許可に対して影響を与えません。このバケットはアクセスコントロールを定義するためだけにポリシーを使用します。

デフォルトでは、ACL は無効になっています。Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

ACL の有効化

- 希望するバケット所有者 — バケット所有者は、他のアカウントが `bucket-owner-full-control` 既定 ACL でバケットに書き込む新しいオブジェクトを所有し、完全にコントロールできます。

  [バケット所有者を推奨] 設定を適用して、すべての Amazon S3 アップロードに `bucket-owner-full-control` 既定 ACL を含めることを要求する場合は、この ACL を使用するオブジェクトアップロードのみを許可するバケットポリシーを追加できます。
- オブジェクトライター — オブジェクトをアップロードする AWS アカウント は、そのオブジェクトを所有し、そのオブジェクトを完全にコントロールし、ACL を介して他のユーザーにそのオブジェクトへのアクセス権を付与できます。

> **Note**
>
> デフォルト設定は [バケット所有者の強制] です。デフォルト設定を適用して ACL を無効のままにするのに必要なのは、`s3:CreateBucket` アクセス許可のみです。ACL を有効にするためには、`s3:PutBucketOwnershipControls` アクセス許可が必要です。

10. [このバケットのパブリックアクセスブロック設定] で、バケットに適用するブロックパブリックアクセス設定を選択します。

    デフォルトでは、4 つすべての [パブリックアクセスをブロック] 設定が有効になっています。特定のユースケースでオフにする必要のある設定が 1 つ以上あることがわかっている場合を除き、すべての設定を有効にしておくことをお勧めします。ブロックパブリックアクセスの詳細については、Amazon S3 ストレージへのパブリックアクセスのブロック を参照してください。

> **Note**
>
> すべての [パブリックアクセスをブロック] 設定を有効にするのに必要なのは、`s3:CreateBucket` アクセス許可のみです。[パブリックアクセスをブロック] 設定をオフにするには、`s3:PutBucketPublicAccessBlock` アクセス許可が必要です。

11. (オプション) [Bucket Versioning] (バケットバージョニング) では、オブジェクトのバリアントをバケットに保持するかどうかを選択できます。バージョニングの詳細については、S3 バケットでのバージョニングの使用 を参照してください。

    バケットのバージョニングを無効または有効にするには、[Disable] (無効化) または [Enable] (有効化) を選択します。

12. (オプション) [Tags] (タグ) で、バケットにタグを追加することを選択できます。タグは、ストレージを分類するために使用できます。

    (オプション) タグを追加するには、[Key] (キー) を入力してから、オプションの [Value] (値) を入力し、[Add Tag] (タグの追加) を選択します。

13. [Default encryption (デフォルトの暗号化)] で、[Edit (編集)] を選択します。

14. デフォルトの暗号化を設定するには、[暗号化タイプ] で次のいずれかを選択します。

    - Amazon S3 マネージドキー (SSE-S3)
    - AWS Key Management Service キー (SSE-KMS)

    > **Important**
    >
    > デフォルト暗号化設定に SSE-KMS オプションを使用する場合、AWS KMS の 1 秒あたりのリクエスト (RPS) 制限が適用されます。AWS KMS クォータの詳細およびクォータの引き上げをリクエストする方法については、AWS Key Management Service デベロッパーガイドの「クォータ」を参照してください。

    バケットと新しいオブジェクトは、暗号化設定の基本レベルとして Amazon S3 マネージドキーを使用したサーバー側の暗号化で暗号化されます。デフォルトの暗号化の詳細については、[Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定] を参照してください。

Amazon S3 のサーバー側の暗号化を使用してデータを暗号化する方法の詳細については、Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) を参照してください。

15. [AWS Key Management Service キー (SSE-KMS)] を選択した場合は、以下の操作を実行します。

    a. [AWS KMS キー] で、次のいずれかの方法で KMS キーを指定します。

    - 使用可能な KMS キーのリストから選択するには、[AWS KMS keys から選択する] を選択し、使用可能なキーのリストから自分の KMS キーを選択します。

      AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がこのリストに表示されます。カスタマーマネージドキーの詳細については、AWS Key Management Service デベロッパーガイドの「カスタマーキーと AWS キー」を参照してください。

    - KMS キー ARN を入力するには、[AWS KMS key ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。
    - AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

      AWS KMS key の作成の詳細については、AWS Key Management Service デベロッパーガイドのキーの作成を参照してください。

> **Important**
>
> バケットと同じ AWS リージョン で使用可能な KMS キーのみを使用できます。Amazon S3 コンソールには、バケットと同じリージョンで最初の 100 個の KMS キーしか表示されません。リストに存在しない KMS キーを使用するには、KMS キー ARN を入力する必要があります。別のアカウントが所有している KMS キーを使用する場合は、まずそのキーを使用するアクセス許可が必要であり、次に KMS キー ARN を入力する必要があります。KMS キーのクロスアカウント権限の詳細については、AWS Key Management Service デベロッパーガイドの「他のアカウントで使用できる KMS キーを作成する」を参照してください。SSE-KMS に関する詳細は、「AWS KMS (SSE-KMS) によるサーバー側の暗号化の指定」を参照してください。
> Amazon S3 でサーバー側の暗号化に AWS KMS key を使用する場合は、対称暗号化 KMS キーを選択する必要があります。Amazon S3 では、対称暗号化 KMS キーのみがサポートされ、非対称暗号化 KMS キーはサポートされません。詳細

については、AWS Key Management Service デベロッパーガイドの「Identifying symmetric and asymmetric KMS keys」(対称および非対称 KMS キーの識別) を参照してください。

AWS KMS key の作成の詳細については、AWS Key Management Service デベロッパーガイドのキーの作成を参照してください。Amazon S3 での AWS KMS の使用に関する詳細は、AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用 を参照してください。

b. SSE-KMS でデフォルトの暗号化を使用するようにバケットを設定する場合は、S3 バケットキーを有効にすることもできます。S3 バケットキーは、Amazon S3 から AWS KMS へのリクエストトラフィックを減らし、暗号化のコストを削減します。詳細については、「Amazon S3 バケットキーを使用した SSE-KMS のコストの削減」を参照してください。

   S3 バケットキーを使用するには、[バケットキー] で [有効化] を選択します。

16. (オプション) S3 オブジェクトロックを有効にする場合は、次の手順に従います。

a. [詳細設定] を選択します。

> **Important**
>
> バケットに対してオブジェクトロックを有効にすると、バージョニングも有効になります。有効にした後、オブジェクトロックのデフォルト保持設定およびリーガルホールド設定を指定し、新しいオブジェクトを削除または上書きしないようにする必要があります。

b. オブジェクトロックを有効にする場合は、[Enable] (有効化) を選択し、表示される警告を読んだうえで承認します。

詳細については、「S3 オブジェクトロックの使用」を参照してください。

> **Note**
>
> オブジェクトロックが有効なバケットを作成するには、`s3:CreateBucket`、`s3:PutBucketVersioning`、および `s3:PutBucketObjectLockConfiguration` の許可が必要です。

17. [Create bucket] (バケットの作成) をクリックします。

Amazon S3 にバケットが作成されました。

次のステップ

オブジェクトをバケットに追加するには、「ステップ 2: バケットにオブジェクトをアップロードする」を参照してください。

ステップ 2: バケットにオブジェクトをアップロードする

Amazon S3 でバケットを作成すると、オブジェクトをバケットにアップロードする準備が整います。オブジェクトは、テキストファイル、写真、ビデオなど、どのような種類のファイルでも可能です。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

オブジェクトをバケットにアップロードするには

1. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets (バケット)] リストで、オブジェクトのアップロード先のバケットの名前を選択します。
3. バケットの [Objects (オブジェクト)] タブで、[Upload (アップロード)] を選択します。
4. [Files and Folders (ファイルとフォルダ)] で、[Add files (ファイルを追加)] を選択します。
5. アップロードするファイルを選択し、続いて [Open (オープン)] を選択します。
6. [アップロード] を選択します。

オブジェクトがバケットに正常にアップロードされました。

次のステップ

オブジェクトを表示するには、「ステップ 3: オブジェクトをダウンロードする」を参照してください。

ステップ 3: オブジェクトをダウンロードする

バケットにオブジェクトをアップロード後、オブジェクトに関する情報を表示し、ローカルコンピュータにそのオブジェクトをダウンロードできます。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

S3 コンソールの使用

このセクションでは、Amazon S3 コンソールを使用して S3 バケットからオブジェクトをダウンロードする方法について説明します。

> **Note**
>
> - 一度にダウンロードできるオブジェクトは 1 つだけです。
> - Amazon S3 コンソールを使用して、キー名がピリオド (.) で終わるオブジェクトをダウンロードすると、ダウンロードしたオブジェクトのキー名からピリオドが削除されます。ダウンロードしたオブジェクトの名前の末尾のピリオドを保持するには、AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 REST API を使用する必要があります。

S3 バケットからオブジェクトをダウンロードするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets] (バケット) リストで、オブジェクトのダウンロード元になるバケット名を選択します。
3. 次のいずれかの方法で、S3 バケットからオブジェクトをダウンロードできます。

- オブジェクトの横にあるチェックボックスを選択し、[ダウンロード] を選択します。オブジェクトを特定のフォルダにダウンロードする場合は、[アクション] メニューの [名前を付けてダウンロード] を選択します。
- オブジェクトの特定のバージョンをダウンロードする場合は、[バージョンの表示] ボタンを選択します。目的のオブジェクトのバージョンの横にあるチェックボックスをオンにして、[ダウンロード] を選択します。オブジェクトを特定のフォルダにダウンロードする場合は、[アクション] メニューの [名前を付けてダウンロード] を選択します。

オブジェクトが正常にダウンロードされました。

次のステップ

Amazon S3 内でオブジェクトをコピーして貼り付ける方法については、「ステップ 4: オブジェクトをフォルダにコピーする」を参照してください。

ステップ 4: オブジェクトをフォルダにコピーする

バケットにオブジェクトを追加し、そのオブジェクトをダウンロードできました。ここで、フォルダを作成し、オブジェクトをコピーしてフォルダに貼り付けます。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

オブジェクトをフォルダにコピーするには

1. [Buckets] (バケット) リストで、バケット名を選択します。
2. [Create folder] (フォルダの作成) を選択して、新しいフォルダを設定します。
   a. フォルダ名 (`favorite-pics` など) を入力します。
   b. 暗号化設定フォルダでは、[Disable] (無効) を選択します。
   c. [Save] (保存) を選択します。
3. コピーするオブジェクトを含む Amazon S3 バケットまたはフォルダに移動します。

4. コピーするオブジェクトの名前の左にあるチェックボックスをオンにします。
5. [Actions (アクション)] を選択し、表示されるオプションのリストから [Copy (コピー)] を選択します。

   または、右上のオプションから [Copy (コピー)] を選択します。
6. コピー先フォルダを選択します。

   a. Browse S3 (S3 の参照) を選択します。
   b. フォルダ名の左にあるオプションボタンを選択します。

      フォルダ内に移動し、コピー先としてサブフォルダを選択するには、フォルダ名を選択します。
   c. [コピー先の選択] を選択します。

   コピー先フォルダへのパスが [コピー先] ボックスに表示されます。[コピー先] には、s3://*bucket-name*/*folder-name*/ などコピー先のパスを入力してもかまいません。
7. 右下の [コピー] を選択します。

   Amazon S3 によってオブジェクトが送信先フォルダにコピーされます。

次のステップ

Amazon S3 でオブジェクトとバケットを削除するときは、「ステップ 5: オブジェクトとバケットを削除する」を参照してください。

ステップ 5: オブジェクトとバケットを削除する

オブジェクトまたはバケットが不要になった場合は、それ以上の料金が発生しないように、オブジェクトまたはバケットを削除することをお勧めします。この開始方法のチュートリアルを演習として完了し、バケットまたはオブジェクトを使用する予定がないときは、料金が発生しないようにするため、バケットを削除することをお勧めします。

バケットを削除する前に、バケットを空にするか、バケット内のオブジェクトを削除します。オブジェクトとバケットを削除すると、それらは使用できなくなります。

引き続き同じバケット名を使用したいときは、オブジェクトを削除するか、バケットを空にすることをお勧めします。ただし、バケットは削除しないでください。バケットを削除すると、その名前は再

利用できるようになります。ただし、バケットを再利用する前に、別の AWS アカウント で同じ名前のバケットが作成される可能性があります。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

トピック

オブジェクトの削除

バケットからすべてのオブジェクトを空にせずに、削除するオブジェクトを選択する場合は、オブジェクトを削除できます。

1. [Buckets (バケット)] リストで、オブジェクトを削除するバケットの名前を選択します。
2. 削除するオブジェクトを選択します。
3. または、右上のオプションから [削除] を選択します。
4. [オブジェクトの削除] ページで、**`delete`** を入力して、オブジェクトの削除を確認します。
5. [Delete objects] (オブジェクトの削除) を選択します。

バケットを空にする

バケットを削除する場合は、まずバケットを空にする必要があります。これにより、バケット内のすべてのオブジェクトが削除されます。

バケットを空にするには

1. [Buckets (バケット)] リストで、空にするバケットを選択し、[Empty (空にする)] を選択します。
2. バケットを空にして、バケット内のすべてのオブジェクトを削除することを確認するには、[Empty bucket] (バケットを空にする) で **`permanently delete`** を入力します。

> **Important**
>
> バケットを空にすると、元に戻すことはできません。バケットを空にするアクションの実行中にバケットに追加されたオブジェクトは削除されます。

3. バケットを空にしてバケット内のすべてのオブジェクトを削除するには、[Empty (空にする)] を選択します。

   [Empty bucket: Status (バケットを空にする: ステータス)] ページが開き、失敗したオブジェクトの削除と成功したオブジェクトの削除の概要を確認できます。
4. バケットリストに戻るには、[Exit (終了)] を選択します。

バケットの削除

バケットを空にするか、バケットからすべてのオブジェクトを削除した後、バケットを削除できます。

1. バケットを削除するには、[Buckets (バケット)] リストでバケットを選択します。
2. [Delete] (削除) をクリックします。
3. 削除を確認するには、[Delete bucket](バケットの削除) でバケットの名前を入力します。

> **Important**
>
> バケットを削除すると、元に戻すことはできません。バケット名は一意です。バケットを削除すると、別の AWS ユーザーがその名前を使用できます。同じバケット名を引き続き使用する場合は、バケットを削除しないでください。代わりに、バケットを空にして保管しておきます。

4. バケットを削除するには、[Delete bucket (バケットの削除)] を選択します。

次のステップ

上記の例で、いくつかの基本的な Amazon S3 のタスクをどのように実行するかを学びました。

次のトピックでは、Amazon S3 をより深く理解して、アプリケーションに実装できるようにするために使用できるラーニングパスについて説明します。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

トピック

一般的なユースケースを理解する

Amazon S3 を使用して、お客様固有のユースケースをサポートすることができます。AWSソリューションライブラリおよびAWS ブログは、ユースケース固有の情報とチュートリアルを提供します。Amazon S3 の一般的ユースケースは以下のとおりです。

- バックアップおよびストレージ – Amazon S3 の ストレージ管理機能を使用して、コストの管理、規制要件への対応、レイテンシーの短縮、コンプライアンス要件のためのデータの複数の異なるコピーの保存を行います。
- アプリケーションホスティング – 信頼性と拡張性が高く、低コストのウェブアプリケーションをデプロイ、インストール、管理できます。例えば、静的ウェブサイトホスティング用に Amazon S3 バケット を設定することができます。詳細については、「Amazon S3 を使用して静的ウェブサイトをホスティングする」を参照してください。
- メディアホスティング – 動画、写真、音楽のアップロードとダウンロードをホスティングし、高い可用性を備えるインフラストラクチャを構築します。
- ソフトウェア配信 – 顧客がダウンロードできるソフトウェアアプリケーションをホスティングします。

バケットとオブジェクトへのアクセスをコントロール

Amazon S3 には、さまざまなセキュリティ機能とツールが用意されています。概要については、「アクセス管理」を参照してください。

デフォルトでは、S3 バケットとオブジェクトはプライベートです。作成した S3 リソースにのみアクセスできます。以下の機能を使用して、特定のユースケースをサポートする詳細なリソース許可を付与したり、Amazon S3 リソースの許可を監査したりできます。

- S3 パブリックアクセスをブロック – S3 バケットおよびオブジェクトへのパブリックアクセスをブロックします。デフォルトでは、[パブリックアクセスをブロック] 設定はバケットレベルで有効になっています。
- AWS Identity and Access Management (IAM) ID — IAM または AWS IAM Identity Center を使用して AWS アカウントで IAM ID を作成し、Amazon S3 のリソースへのアクセスを管理します。例えば、Amazon S3 で IAM を使用して、AWS アカウント が所有する Amazon S3 バケットの特定の部分に対するユーザーまたはユーザーグループのアクセス権のタイプをコントロールすることができます。IAM ID およびベストプラクティスの詳細については、IAM ユーザーガイドの「IAM ID (ユーザー、ユーザーグループ、ロール)」を参照してください。
- バケットポリシー – IAM ベースのポリシー言語を使用して、S3 バケットとその中のオブジェクトに対するリソースベースの許可を設定します。
- アクセスコントロールリスト (ACL) – 個々のバケットおよびオブジェクトに対する読み取りおよび書き込みの許可を、承認されたユーザーに付与します。原則として、アクセスコントロールには ACL ではなく S3 リソースベースのポリシー (バケットポリシーとアクセスポイントポリシー) または IAM ユーザーポリシーを使用することをお勧めします。ポリシーとは、よりシンプルで柔軟なアクセス制御のオプションです。バケットポリシーとアクセスポイントポリシーを使用すると、Amazon S3 リソースに対するすべてのリクエストに広く適用されるルールを定義できます。リソースベースのポリシーまたは IAM ユーザーポリシーの代わりに ACL を使用する場合の特定のケースの詳細については、「Amazon S3 用 Identity and Access Management」を参照してください。
- S3 オブジェクト所有権 - バケット内のすべてのオブジェクトの所有権を取得し、Amazon S3 に保存されているデータのアクセス管理を簡素化します。S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、ACL を無効または有効にするのに使用できます。デフォルトでは、ACL は無効になっています。ACL を無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。
- IAM Access Analyzer for S3 – S3 バケットアクセスポリシーを評価およびモニタリングし、ポリシーが S3 リソースへの意図したアクセスのみを提供することを確認します。

ストレージの管理と監視

- ストレージの管理 – Amazon S3 でバケットを作成してオブジェクトをアップロードすると、オブジェクトストレージを管理できるようになります。例えば、災害対策に S3 バージョニングと S3 レプリケーション、ストレージコストを管理する S3 ライフサイクル、コンプライアンス要件を満たすために S3 オブジェクトロックを使用できます。
- ストレージのモニタリング – モニタリングは、Amazon S3 および AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。ストレージのアクティビティとコストを監視できます。また、マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできるように、AWS ソリューションのすべての部分からモニタリングデータを収集することをお勧めします。
- Amazon S3 の分析とインサイト – Amazon S3 の分析とインサイトを使用して、ストレージの使用状況を把握、分析、最適化することができます。例えば、Amazon S3 Storage Lens を使用して、ストレージの把握、分析、最適化を行います。S3 Storage Lens には、組織全体、特定のアカウント、リージョン、バケット、またはプレフィックスのデータを集約するための 29 以上の使用状況およびアクティビティに関するメトリクスとインタラクティブなダッシュボードが用意されています。ストレージクラス分析を使用してストレージアクセスパターンを分析し、データをよりコスト効率の高いストレージクラスに移動するタイミングを決定します。

Amazon S3 を使用した開発

Amazon S3 は REST サービスです。REST API または、基礎となる Amazon S3 REST API をラップする AWS SDK ライブラリを使用して、Amazon S3 にリクエストを送信できます。これにより、プログラミング作業が簡易になります。また、AWS Command Line Interface (AWS CLI) を使用して、Amazon S3 API 呼び出しを行います。詳細については、「リクエストの実行」を参照してください。

Amazon S3 REST API は、Amazon S3 に対する HTTP インターフェイスです。REST API を使用すると、標準 HTTP リクエストを使用してバケットとオブジェクトを作成、取得、および削除することができます。REST API を使用する場合、HTTP をサポートする任意のツールキットを使用できます。匿名で読み取り可能なオブジェクトであれば、ブラウザを使用して取得することもできます。詳細については、「REST API を使用した Amazon S3 での開発」を参照してください。

お客様が選択した言語を使用してアプリケーションを構築できるよう、以下のリソースを提供しています。

AWS CLI

AWS CLI を使用して、Amazon S3 の機能にアクセスできます。AWS CLI をダウンロードして設定するには、AWS CLI を使用した Amazon S3 での開発 を参照してください。

AWS CLI は Amazon S3 にアクセスするための 2 つの階層のコマンドとして、高レベル (s3) コマンドと APIレベル (s3api および s3control) コマンドを提供しています。高レベル S3 コマンドは、オブジェクトおよびバケットの作成、操作、削除など、一般的なタスクの実行を簡素化します。s3api および s3control コマンドは、すべての Amazon S3 API オペレーションへの直接アクセスを提供します。これにより、高レベルコマンドだけでは不可能なアドバンスドオペレーションを実行することができます。

Amazon S3 AWS CLI コマンドのリストについては、s3、s3api、および s3control を参照してください。

AWS SDK と Explorer

Amazon S3 でのアプリケーション開発に AWS SDK を使用できます。AWS SDK は、基盤となる REST API をラップして、プログラミング作業を簡素化します。AWS を使用して、接続されるモバイルおよびウェブアプリケーションを構築するために、AWS Mobile SDK と Amplify JavaScript ライブラリも用意されています。

AWS SDK に加え、Visual Studio および Eclipse for Java IDE で使用できる AWS Explorer も提供されています。この場合、SDK と Explorer が、AWS ツールキットとしてバンドルされて提供されます。

詳細については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。

サンプルコードおよびライブラリ

AWS デベロッパーセンターおよびAWS サンプルコードカタログには、Amazon S3 用に特別に作成されたサンプルコードとライブラリが用意されています。これらのサンプルコードを、Amazon S3 API の実装方法を理解する教材としてご利用いただけます。また、Amazon Simple Storage Service API リファレンスを参照して、Amazon S3 API オペレーションについて詳しく理解してください。

チュートリアルから学ぶ

ステップバイステップのチュートリアルを開始して、Amazon S3 の詳細について学びます。これらのチュートリアルはラボ型環境向けであり、架空の企業名やユーザー名などを使用しています。目的は、一般的なガイダンスを提供することです。お客様の組織環境に固有のニーズを満たすかどうかの十分な確認や調整をすることなく、本番環境で直接使用するためのものではありません。

使用開始

- チュートリアル: Amazon S3 によるファイルの保存と取得
- チュートリアル: S3 Intelligent-Tiering を使用した開始方法
- チュートリアル: Amazon S3 Glacier ストレージクラスを使用した開始方法

ストレージコストの最適化

- チュートリアル: S3 Intelligent-Tiering を使用した開始方法
- チュートリアル: Amazon S3 Glacier ストレージクラスを使用した開始方法
- チュートリアル: S3 ストレージレンズによるコストの最適化と使用状況の可視化

ストレージの管理

- チュートリアル: Amazon S3 マルチリージョンアクセスポイントの使用の開始方法
- チュートリアル: S3 バッチレプリケーションによる Amazon S3 バケット内の既存のオブジェクトのレプリケーション

動画とウェブサイトのホスティング

- チュートリアル: Amazon S3、Amazon CloudFront、Amazon Route 53 を使用したオンデマンドストリーミング動画のホスティング。
- チュートリアル: Amazon S3 での静的ウェブサイトの設定
- チュートリアル: Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定

データの処理

- チュートリアル: S3 Object Lambda を使用したアプリケーションのデータの変換
- チュートリアル: S3 Object Lambda と Amazon Comprehend を使用した PII データの検出と編集
- チュートリアル: S3 Object Lambda を使用して、取得時に画像に動的に透かしを入れる
- チュートリアル: S3 バッチオペレーション、AWS Lambda、および AWS Elemental MediaConvert を使用した動画のバッチトランスコーディング

データの保護

- チュートリアル: チェックサムを追加して Amazon S3 のデータの整合性をチェックする
- チュートリアル: S3 レプリケーションを使用して AWS リージョン内およびリージョン間でデータをレプリケートする
- チュートリアル: S3 バージョニング、S3 オブジェクトロック、S3 レプリケーションを使用して、Amazon S3 上のデータを予期しない削除やアプリケーションのバグから保護する
- チュートリアル: S3 バッチレプリケーションによる Amazon S3 バケット内の既存のオブジェクトのレプリケーション

トレーニングとサポートを見る

AWS のエキスパートから学び、スキルを向上させ、目的を達成するために工キスパートの支援を受けることができます。

- トレーニング – トレーニングリソースは、Amazon S3 を学習するための実践的なアプローチです。詳細については、「AWS トレーニングと認定」および「AWS オンラインテクニカルトーク」を参照してください。
- ディスカッションフォーラム – フォーラムでは、投稿を確認して、Amazon S3 で何ができるのか、何ができないのかを理解することができます。質問を投稿することもできます。詳細については、「ディスカッションフォーラム」を参照してください。
- 技術サポート – さらにご質問がある場合は、技術サポートまでご連絡ください。

チュートリアル

次のチュートリアルでは、Amazon S3 の一般的なタスクにおけるエンドツーエンドの一連の手順について説明します。これらのチュートリアルはラボ型環境向けであり、架空の企業名やユーザー名などを使用しています。目的は、一般的なガイダンスを提供することです。お客様の組織環境に固有のニーズを満たすかどうかの十分な確認や調整をすることなく、本番環境で直接使用するためのものではありません。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

使用開始

- チュートリアル: Amazon S3 によるファイルの保存と取得
- チュートリアル: S3 Intelligent-Tiering を使用した開始方法
- チュートリアル: Amazon S3 Glacier ストレージクラスを使用した開始方法

ストレージコストの最適化

- チュートリアル: S3 Intelligent-Tiering を使用した開始方法
- チュートリアル: Amazon S3 Glacier ストレージクラスを使用した開始方法
- チュートリアル: S3 ストレージレンズによるコストの最適化と使用状況の可視化

ストレージの管理

- チュートリアル: Amazon S3 マルチリージョンアクセスポイントの使用の開始方法
- チュートリアル: S3 バッチレプリケーションによる Amazon S3 バケット内の既存のオブジェクトのレプリケーション

動画とウェブサイトのホスティング

- チュートリアル: Amazon S3、Amazon CloudFront、Amazon Route 53 を使用したオンデマンドストリーミング動画のホスティング。
- チュートリアル: Amazon S3 での静的ウェブサイトの設定
- チュートリアル: Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定

データの処理

- チュートリアル: S3 Object Lambda を使用したアプリケーションのデータの変換
- チュートリアル: S3 Object Lambda と Amazon Comprehend を使用した PII データの検出と編集
- チュートリアル: S3 Object Lambda を使用して、取得時に画像に動的に透かしを入れる
- チュートリアル: S3 バッチオペレーション、AWS Lambda、および AWS Elemental MediaConvert を使用した動画のバッチトランスコーディング

データの保護

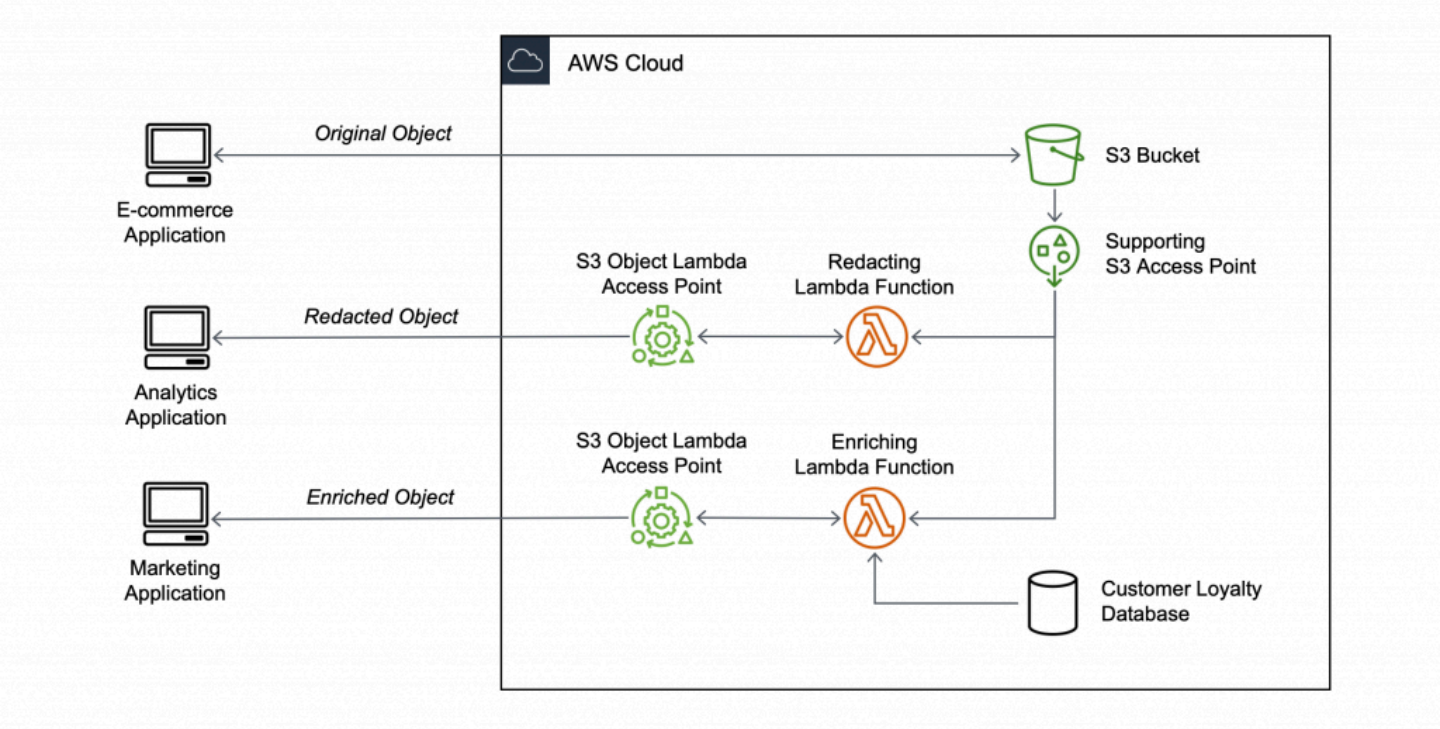
- チュートリアル: チェックサムを追加して Amazon S3 のデータの整合性をチェックする
- チュートリアル: S3 レプリケーションを使用して AWS リージョン内およびリージョン間でデータをレプリケートする
- チュートリアル: S3 バージョニング、S3 オブジェクトロック、S3 レプリケーションを使用して、Amazon S3 上のデータを予期しない削除やアプリケーションのバグから保護する
- チュートリアル: S3 バッチレプリケーションによる Amazon S3 バケット内の既存のオブジェクトのレプリケーション

チュートリアル: S3 Object Lambda を使用したアプリケーションのデータの変換

Amazon S3 にデータを保管すると、複数のアプリケーションで使用するために簡単にデータを共有できます。ただし、各アプリケーションには固有のデータ形式の要件があり、特定のユースケースでデータの変更や処理が必要になる場合があります。例えば、e コマースアプリケーションによって作成されたデータセットには、個人を特定できる情報 (PII) が含まれている場合があります。分析のために同じデータが処理される場合、この PII は必要ないため、編集する必要があります。ただし、同

じデータセットをマーケティングキャンペーンに使用する場合は、顧客ロイヤルティデータベースからの情報など、追加の詳細でデータを充実させる必要がある場合があります。

S3 Object Lambda を使用すると、独自のコードを追加して、データがアプリケーションに返される前に S3 からのデータを処理できます。具体的には、AWS Lambda 関数を設定して、S3 Object Lambda アクセスポイントにアタッチします。アプリケーションが S3 Object Lambda アクセスポイントを介して 標準 S3 GET リクエスト を送信すると、指定された Lambda 関数が呼び出され、サポートする S3 アクセスポイントを介して S3 バケットから取得されたデータが処理されます。その後、S3 Object Lambda アクセスポイントは、変換された結果をアプリケーションに返します。独自のカスタム Lambda 関数を作成して実行し、S3 Object Lambda データ変換を特定のユースケースに合わせて調整できます。すべて、アプリケーションの変更は不要です。



目的

このチュートリアルでは、標準の S3 GET リクエストにカスタムコードを追加して、リクエストしたクライアントまたはアプリケーションのニーズに合うように、S3 から取得したリクエストされたオブジェクトを変更する方法を学習します。具体的には、S3 に保存されている元のオブジェクトのすべてのテキストを S3 Object Lambda を使用して大文字に変換する方法を学習します。

トピック

- 前提条件
- ステップ 1: S3 バケットを作成する。

前提条件

このチュートリアルを開始する前に、適切な許可が付与された AWS Identity and Access Management (IAM) ユーザーとしてサインインできる AWS アカウントが必要です。また、バージョン 3.8 以降の Python をインストールする必要があります。

サブステップ

- AWS アカウントでアクセス許可を持つ IAM ユーザーを作成します (コンソール)。
- ローカルマシンに Python 3.8 以降をインストールする

AWS アカウントでアクセス許可を持つ IAM ユーザーを作成します (コンソール)。

チュートリアル用の IAM ユーザーを作成できます。このチュートリアルを完了するには、IAM ユーザーが次の IAM ポリシーをアタッチして、関連する AWS リソースにアクセスし、特定のアクションを実行する必要があります。IAM ユーザーを作成する方法の詳細については、「IAM ユーザーガイド」の「IAM ユーザーの作成 (コンソール)」を参照してください。

IAM ユーザーには次のポリシーが必要です。

- AmazonS3FullAccess – Object Lambda アクセスポイントを作成および使用するための許可を含む、すべての Amazon S3 アクションに対する許可を付与します。
- AWSLambda_FullAccess – すべての Lambda アクションに許可を付与します。
- IAMFullAccess – すべての IAM アクションに許可を付与します。
- IAMAccessAnalyzerReadOnlyAccess – IAM Access Analyzer によって提供されるすべてのアクセス情報を読み取る許可を付与します。
- CloudWatchLogsFullAccess – CloudWatch Logs へのフルアクセスを付与します。

> **Note**
>
> このチュートリアルでは、わかりやすいように IAM ユーザーを作成して使用します。このチュートリアルを完了したら、忘れずに「IAM ロールを削除する」を行います。本番環境で使用する場合は、IAM ユーザーガイドの「IAM でのセキュリティのベストプラクティス」に従うことをお勧めします。ベストプラクティスでは、人間のユーザーと ID プロバイダーとのフェデレーションにより、一時的な認証情報を使用して AWS にアクセスする必要があります。追加のベストプラクティスとして、ワークロードでは一時的な認証情報で IAM ロールを使用し、AWS にアクセス必要があります。AWS IAM Identity Center を使用して一時的な認証情報を持つユーザーを作成する方法については、「AWS IAM Identity Center ユーザーガイド」の「開始方法」を参照してください。
> このチュートリアルでは、フルアクセス権を持つ AWS マネージドポリシーも使用します。実稼働環境では、代わりに、セキュリティのベストプラクティスに従って、ユースケースに必要な最小限の許可のみを付与することをお勧めします。

ローカルマシンに Python 3.8 以降をインストールする

次の手順に従って、Python 3.8 以降をローカルマシンにインストールします。インストール手順については、Python の Beginner Guide で「Python のダウンロード」ページを参照してください。

1. ローカルターミナルまたはシェルを開き、次のコマンドを実行して、Python がすでにインストールされているかどうか、インストールされている場合は、どのバージョンがインストールされているかを確認します。

   ```
   python --version
   ```

2. Python 3.8 以降をお持ちでない場合は、ローカルマシンに適した Python 3.8 以降の公式インストーラをダウンロードします。
3. ダウンロードしたファイルをダブルクリックしてインストーラを実行し、手順に従ってインストールを完了します。

   Windows ユーザーの場合、インストールウィザードで [Python 3.X を PATH に追加] を選択してから、[今すぐ、インストール] を選択します。
4. ターミナルを閉じて再度開いて、再起動します。
5. 次のコマンドを実行して、Python 3.8 以降が正しくインストールされていることを確認します。

macOS ユーザーの場合は、このコマンドを実行します。

```
python3 --version
```

Windows ユーザーの場合は、このコマンドを実行します。

```
python --version
```

6. 次のコマンドを実行して、pip3 パッケージマネージャーがインストールされていることを確認します。コマンド応答に pip のバージョン番号と python 3.8 以降が表示された場合、pip3 パッケージマネージャが正常にインストールされていることを意味します。

```
pip --version
```

ステップ 1: S3 バケットを作成する。

変換する元のデータを保存するバケットを作成します。

バケットを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [Create bucket (バケットの作成)] を選択します。

   [バケットの作成] ページが開きます。
4. [バケット名] に、バケットの名前 (例、**tutorial-bucket**) を入力します。

   Amazon S3 のバケット命名規則の詳細については、「バケットの名前付け」を参照してください。
5. [リージョン] で、バケットを保存する AWS リージョンを選択します。

   バケットのリージョンの詳細については、「バケットの概要」を参照してください。
6. [このバケットのパブリックアクセス設定をブロック] で、デフォルト設定 (ブロックすべてパブリックアクセスが有効) のままであることを確認します。

ユースケースでオフにする必要のある設定が 1 つ以上あることがわかっている場合を除き、すべてのブロックパブリックアクセス設定を有効にしておくことをお勧めします。ブロックパブリックアクセスの詳細については、Amazon S3 ストレージへのパブリックアクセスのブロックを参照してください。

7. 残りの設定はデフォルトのままにしておきます。

   (オプション) 特定のユースケースに追加のバケット設定を設定する場合は、「バケットの作成」を参照してください。

8. [Create bucket (バケットの作成)] を選択します。

ステップ 2: S3 バケットにファイルをアップロード

S3 バケットにテキストファイルをアップロードします。このテキストファイルには、このチュートリアルの後半で大文字に変換する元のデータが含まれています。

例えば、次のテキストが含まれている `tutorial.txt` ファイルをアップロードできます。

```
Amazon S3 Object Lambda Tutorial:
You can add your own code to process data retrieved from S3 before
returning it to an application.
```

バケットにファイルをアップロードするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、ステップ 1 で作成したバケットの名前を選択し (**`tutorial-bucket`** など)、ファイルをアップロードします。
4. バケットの [オブジェクト] タブで、[アップロード] を選択します。
5. [アップロード] ページの [ファイルとフォルダ] の下で、[ファイルを追加] を選択します。
6. アップロードするファイルを選択し、続いて [オープン] を選択します。例えば、前述した `tutorial.txt` ファイルの例をアップロードできます。
7. [アップロード] を選択します。

ステップ 3: S3 アクセスポイントの作成

S3 Object Lambda アクセスポイントを使用して元のデータにアクセスし、変換するには、S3 アクセスポイントを作成し、ステップ 1 で作成した S3 バケットに関連付けます。アクセスポイントは、変換するオブジェクトと同じ AWS リージョン に存在する必要があります。

このチュートリアルの後半では、このアクセスポイントを Object Lambda アクセスポイントのサポートアクセスポイントとして使用します。

アクセスポイントを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[アクセスポイント] を選択します。
3. [アクセスポイント] ページで、[アクセスポイントの作成] を選択します。
4. [アクセスポイント名] フィールドに、アクセスポイントの名前を入力します (**`tutorial-access-point`** など)。

   アクセスポイントの名前付けの詳細については、「Amazon S3 アクセスポイントの命名規則」を参照してください。
5. [バケット名] フィールドに、ステップ 1 で作成したバケットの名前を入力します(**`tutorial-bucket`** など)。S3 は、アクセスポイントをこのバケットにアタッチします。

   (オプション) [S3 を参照] を選択して、アカウント内のバケットを参照および検索できます。[S3 を参照] を選択した場合は、目的のバケットを選択し、[パスの選択] を選択して、[バケット名] フィールドにバケットの名前を入力します。
6. [ネットワークオリジン] で、[インターネット] を選択します。

   アクセスポイントのネットワークオリジンの詳細については、「Virtual Private Cloud に制限されたアクセスポイントの作成」を参照してください。
7. デフォルトでは、アクセスポイントに対してすべてのブロックパブリックアクセス設定が有効になります。パブリックアクセスブロック設定は、すべてを有効にしておくことを推奨します。

   詳細については、「アクセスポイントへのパブリックアクセスの管理」を参照してください。
8. その他のすべてのアクセスポイント設定で、デフォルトの設定を保持します。

   (オプション) アクセスポイントの設定は、ユースケースをサポートするように変更できます。このチュートリアルでは、デフォルトの設定を保持することをお勧めします。

(オプション) アクセスポイントへのアクセスを管理する必要がある場合は、アクセスポイントポリシーを指定できます。詳細については、「アクセスポイントポリシーの例」を参照してください。

9. [アクセスポイントを作成] を選択します。

ステップ 4: Lambda 関数を作成する

元のデータを変換するには、S3 Object Lambda アクセスポイントで使用する Lambda 関数を作成します。

サブステップ

- Lambda 関数コードを記述し、仮想環境でデプロイパッケージを作成する
- 実行ロール (コンソール) を使用した Lambda 関数の作成
- .zip ファイルアーカイブを使用して Lambda 関数コードをデプロイし、Lambda 関数を設定 (コンソール)

Lambda 関数コードを記述し、仮想環境でデプロイパッケージを作成する

1. ローカルマシンで、このチュートリアルの後半で使用する仮想環境に対して、フォルダ名が `object-lambda` であるフォルダを作成します。
2. `object-lambda` フォルダで、元のオブジェクトのすべてのテキストを大文字に変更する Lambda 関数を含むファイルを作成します。例えば、Python で記述された次の関数を使用できます。この関数は、`transform.py` という名前のファイルに保存してください。

```
import boto3
import requests
from botocore.config import Config

# This function capitalizes all text in the original object
def lambda_handler(event, context):
    object_context = event["getObjectContext"]
    # Get the presigned URL to fetch the requested original object
    # from S3
    s3_url = object_context["inputS3Url"]
    # Extract the route and request token from the input context
    request_route = object_context["outputRoute"]
    request_token = object_context["outputToken"]
```

```
    # Get the original S3 object using the presigned URL
    response = requests.get(s3_url)
    original_object = response.content.decode("utf-8")

    # Transform all text in the original object to uppercase
    # You can replace it with your custom code based on your use case
    transformed_object = original_object.upper()

    # Write object back to S3 Object Lambda
    s3 = boto3.client('s3', config=Config(signature_version='s3v4'))
    # The WriteGetObjectResponse API sends the transformed data
    # back to S3 Object Lambda and then to the user
    s3.write_get_object_response(
        Body=transformed_object,
        RequestRoute=request_route,
        RequestToken=request_token)

    # Exit the Lambda function: return the status code
    return {'status_code': 200}
```

> **Note**
>
> 前述の例の Lambda 関数は、リクエストされたオブジェクト全体をメモリにロードしてから、変換してクライアントに返します。または、オブジェクト全体をメモリにロードしないようにするために、S3 からオブジェクトをストリーミングすることもできます。この方法は、大きなオブジェクトを扱うときに役立ちます。Object Lambda アクセスポイントを使用したレスポンスのストリーミングの詳細については、Lambda での GetObject リクエストの使用 にあるストリーミングの例を参照してください。

S3 Object Lambda アクセスポイントで使用する Lambda 関数を記述する場合、この関数は S3 Object Lambda が Lambda 関数に提供する入力イベントコンテキストに基づいています。イベントコンテキストは、S3 Object Lambda から Lambda に渡されたイベントで行われたリクエストに関する情報を提供します。これには、Lambda 関数の作成に使用するパラメータが含まれています。

前述の Lambda 関数の作成に使用されるフィールドは次のとおりです。

`getObjectContext` のフィールドは、Amazon S3 および S3 Object Lambda への接続に関する入力および出力の詳細を意味します。以下のフィールドがあります。

- `inputS3Url` – Lambda 関数がサポートするアクセスポイントから元のオブジェクトをダウンロードするために使用できる署名付き URL です。Lambda 関数は、署名付き URL を使用することにより、元のオブジェクトを取得するために Amazon S3 の読み取り許可を必要とせず、呼び出しごとに処理されたオブジェクトにのみアクセスできます。
- `outputRoute` – Lambda 関数が変換したオブジェクトを戻すために `WriteGetObjectResponse` を呼び出すときに S3 Object Lambda URL に追加されるルーティングトークン。
- `outputToken` – 変換されたオブジェクトを送り返すときに、`WriteGetObjectResponse` 呼び出しを元の呼び出し元と照合するために S3 Object Lambda によって使用されるトークンです。

イベントコンテキスト内のすべてのフィールドの詳細については、「イベントコンテキストの形式と使用法 および S3 Object Lambda アクセスポイントの Lambda 関数の記述」を参照してください。

3. ローカルターミナルで次のコマンドを入力し、`virtualenv` パッケージをインストールします。

```
python -m pip install virtualenv
```

4. ローカルターミナルで、前に作成した `object-lambda` フォルダを開き、次のコマンドを入力して、`venv` という名前の仮想環境を作成し、初期化します。

```
python -m virtualenv venv
```

5. 仮想環境をアクティブ化するには、次のコマンドを入力して環境のフォルダから `activate` ファイルを実行します。

macOS ユーザーの場合は、このコマンドを実行します。

```
source venv/bin/activate
```

Windows ユーザーの場合は、このコマンドを実行します。

```
.\venv\Scripts\activate
```

ここで、コマンドプロンプトが変わり、仮想環境がアクティブであることを示す (venv) が表示されます。

6. 必要なライブラリをインストールするには、venv 仮想環境で、以下のコマンドを行単位で実行します。

   これらのコマンドは、lambda_handler Lambda 関数の依存関係の更新バージョンをインストールします。これらの依存関係は、AWS SDK for Python (Boto3) とリクエストモジュールです。

   ```
   pip3 install boto3
   ```

   ```
   pip3 install requests
   ```

7. 仮想環境を無効にするには、次のコマンドを実行します。

   ```
   deactivate
   ```

8. インストール済みライブラリを含むデプロイパッケージを object-lambda ディレクトリのルートにある lambda.zip という名前の .zip ファイルで作成し、以下のコマンドをローカルターミナルで行単位で実行します。

   > **Tip**
   >
   > 以下のコマンドは、特定の環境で動作するように調整する必要がある場合があります。例えば、ライブラリは site-packages または dist-packages であり、最初のフォルダはlib または lib64 です。また、python フォルダには、別の Python バージョンで名前が付けられている可能性があります。特定のパッケージを見つけるには、pip show コマンドを使用します。

   macOS ユーザーの場合は、このコマンドを実行します。

   ```
   cd venv/lib/python3.8/site-packages
   ```

```
zip -r ../../../../lambda.zip .
```

Windows ユーザーの場合は、このコマンドを実行します。

```
cd .\venv\Lib\site-packages\
```

```
powershell Compress-Archive * ../../../lambda.zip
```

最後のコマンドは、デプロイメントパッケージを object-lambda ディレクトリのルートに保存します。

9. デプロイパッケージのルートに関数コードファイル transform.py を追加します。

macOS ユーザーの場合は、このコマンドを実行します。

```
cd ../../../../
```

```
zip -g lambda.zip transform.py
```

Windows ユーザーの場合は、このコマンドを実行します。

```
cd ..\..\..\
```

```
powershell Compress-Archive -update transform.py lambda.zip
```

このステップを完了すると、以下のようなディレクトリ構造になります。

```
lambda.zip$
  # transform.py
  # __pycache__
  | boto3/
  # certifi/
  # pip/
  # requests/
  ...
```

実行ロール (コンソール) を使用した Lambda 関数の作成

1. AWS Management Console にサインインして AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. 左側のナビゲーションペインで、[関数] を選択します。
3. [Create function (関数の作成)] を選択します。
4. Author from scratch を選択します。
5. 基本的な情報 で、以下の作業を行います。
   a. [関数名] に「**tutorial-object-lambda-function**」と入力します。
   b. [ランタイム] で、Python 3.8 またはそれ以降のバージョンを選択します。
6. [デフォルト実行ロールの変更] セクションを展開します。[実行ロール] で、[基本的な Lambda 許可で新しいロールを作成] を選択します。

   このチュートリアルのステップ 5 で、AmazonS3ObjectLambdaExecutionRolePolicy をこの Lambda 関数の実行ロールにアタッチします。
7. 残りの設定はデフォルト値のままにしておきます。
8. [Create function (関数の作成)] を選択します。

.zip ファイルアーカイブを使用して Lambda 関数コードをデプロイし、Lambda 関数を設定 (コンソール)

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) で、左のナビゲーションペインの関数を選択します。
2. 前に作成した Lambda 関数を選択します (例、**tutorial-object-lambda-function**)。
3. Lambda 関数の詳細ページで、[コード] を選択します。[コードソース] セクションで、[からアップロード]、[.zip ファイル] の順に選択します。
4. [アップロード] を選択して、ローカルの `.zip` ファイルを選択します。
5. 前に作成した `lambda.zip` ファイルを選択し、[開く] を選択します。
6. [Save] を選択します。
7. [ランタイム設定] セクションで、[編集] を選択します。
8. [ランタイム設定の編集] ページで、ランタイムが Python 3.8 またはそれ以降のバージョンに設定されていることを確認します。

9. Lambda ランタイムに Lambda 関数コード内のどのハンドラメソッドを呼び出すかを指示するには、[ハンドラー] に **`transform.lambda_handler`** を入力します。

   Python で関数を設定するとき、ハンドラー設定の値は、ファイル名と、ハンドラーモジュールの名前を、ドットで区切ったものになります。例えば、`transform.lambda_handler` は、`transform.py` ファイルで定義された `lambda_handler` メソッドを呼び出します。

10. [Save] を選択します。

11. (オプション) Lambda 関数の詳細ページで、[設定] タブを選択します。左のナビゲーションペインで、[一般的な設定] を選択してから、[編集] を選択します。[タイムアウト] フィールドに、**1** 分 **0** 秒と入力します。残りの設定はデフォルト値のままにして、[保存] を選択します。

    タイムアウトは、Lambda で関数が停止するまでに許可される実行時間の長さです。デフォルト値は 3 秒です。S3 Object Lambda によって使用される Lambda 関数の最大持続時間は 60 秒です。料金は、設定されたメモリの量とコードの実行時間に基づいて請求されます。

ステップ 5: Lambda 関数の実行ロールの IAM ポリシーを設定する

Lambda 関数がカスタマイズされたデータと応答ヘッダーを `GetObject` 呼び出し元に提供できるようにするには、Lambda 関数の実行ロールに `WriteGetObjectResponse` API を呼び出すための IAM 許可が必要です。

IAM ポリシーを Lambda 関数ロールにアタッチするには

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) で、左のナビゲーションペインの関数を選択します。
2. ステップ 4 で作成した関数 (たとえば、**`tutorial-object-lambda-function`**) を選択します。
3. Lambda 関数の詳細ページで、[設定] タブを選択してから、左のナビゲーションペインの [許可] を選択します。
4. [実行ロール] で、[ロール名] のリンクを選択します。IAM コンソールが開きます。
5. Lambda 関数の実行ロールの IAM コンソールの [Summary] (概要) ページで、[Permissions] (許可) タブを選択します。[Add Permissions] (許可を追加) メニューから [Attach policies] (ポリシーを添付) を選択します。
6. [許可のアタッチ] ページで、[検索] ボックスに **`AmazonS3ObjectLambdaExecutionRolePolicy`** を入力して、ポリシーのリストをフィルタ

リングします。AmazonS3ObjectLambdaExecutionRolePolicy ポリシーの名前の横にあるチェックボックスを選択します。

7. [ポリシーのアタッチ] を選択します。

ステップ 6: Object Lambda アクセスポイントの作成

S3 Object Lambda アクセスポイントは、S3 GET リクエストから Lambda 関数を直接呼び出す柔軟性を提供し、関数が S3 アクセスポイントから取得したデータを処理できるようにします。S3 Object Lambda アクセスポイントを作成および設定するときは、Lambda が使用するカスタムパラメータとして JSON 形式のイベントコンテキストを呼び出し、提供する Lambda 関数を指定する必要があります。

S3 Object Lambda アクセスポイントを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Object Lambda アクセスポイント] を選択します。
3. [Object Lambda アクセスポイント] ページで、[Object Lambda アクセスポイントの作成] を選択します。
4. Object Lambda アクセスポイント名] に、Object Lambda アクセスポイントに使用する名前を入力します (例、**`tutorial-object-lambda-accesspoint`**)。
5. [サポートするアクセスポイント] で、ステップ 3 で作成した標準アクセスポイントを入力または参照します (**`tutorial-access-point`** など)。次に、[サポートするアクセスポイントの選択] を選択します。
6. S3 API の場合、Lambda 関数が処理する S3 バケットからオブジェクトを取得するには、[GetObject] を選択します。
7. [Lambda 関数の呼び出し] では、このチュートリアルでは、次の 2 つのオプションのいずれかを選択できます。
   - [アカウントの関数から選ぶ] を選択し、[Lambda 関数] ドロップダウンリストから、ステップ 4 で作成した Lambda 関数を選択します (例、**`tutorial-object-lambda-function`**)。
   - [ARN を入力] を選択し、ステップ 4 で作成した Lambda 関数の Amazon リソースネーム (ARN) を入力します。
8. [Lambda 関数のバージョン] で、$LATEST (ステップ 4 で作成した Lambda 関数の最新バージョン) を選択します。

9. (オプション) Lambda 関数で、範囲とパート番号によって GET リクエストを認識および処理する必要がある場合は、[Lambda 関数は、範囲を使用してリクエストをサポート] および [Lambda 関数は、パート番号を使用してリクエストをサポート] を選択します。それ以外の場合は、これらの 2 つのチェックボックスをオフにします。

   S3 Object Lambda で範囲またはパート番号を使用する方法の詳細については、「Range および partNumber ヘッダーの操作」を参照してください。

10. (オプション) [ペイロード - オプション] で JSON テキストを追加して、Lambda 関数に追加情報を提供します。

    ペイロードは、特定の S3 Object Lambda アクセスポイントからのすべての呼び出しに対する入力として、Lambda 関数に提供できるオプションの JSON テキストです。同じ Lambda 関数を呼び出す複数の Object Lambda アクセスポイントの動作をカスタマイズして、異なるパラメータを使用してペイロードを設定できます。これにより、Lambda 関数の柔軟性が向上します。

    ペイロードの詳細については、「イベントコンテキストの形式と使用法」を参照してください。

11. (オプション) [リクエストメトリクス - オプション] で、[無効] または [有効] を選択して、Amazon S3 モニタリングを Object Lambda アクセスポイントに追加します。リクエストメトリクスには、Amazon CloudWatch の標準料金が課金されます。詳細については、「CloudWatch 料金表」を参照してください。

12. [Object Lambda アクセスポイントポリシー-オプション] で、デフォルトの設定を保持します。

    (オプション) リソースポリシーを設定できます。このリソースポリシーは、指定された Object Lambda アクセスポイントを使用する `GetObject` API 許可を付与します。

13. 残りの設定はデフォルト値のままにしておき、[Object Lambda アクセスポイントの作成] を選択します。

ステップ 7: 変換されたデータを表示する

これで、S3 Object Lambda は、ユースケースに合わせてデータを変換する準備が整いました。このチュートリアルでは、S3 Object Lambda はオブジェクト内のすべてのテキストを大文字に変換します。

サブステップ

- S3 Object Lambda アクセスポイントの変換済みデータの表示
- Python スクリプトを実行して、元のデータと変換されたデータを出力する

S3 Object Lambda アクセスポイントの変換済みデータの表示

S3 Object Lambda アクセスポイントを使用してファイルの取得をリクエストすると、S3 Object Lambda への `GetObject` API 呼び出しが行われます。S3 Object Lambda は Lambda 関数を呼び出してデータを変換し、変換されたデータを標準 S3 `GetObject` API への応答として返します。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Object Lambda アクセスポイント] を選択します。
3. [Object Lambda アクセスポイント] ページで、ステップ 6 で作成した S3 Object Lambda アクセスポイントを選択します (例、**`tutorial-object-lambda-accesspoint`**)。
4. S3 Object Lambda アクセスポイントの [オブジェクト] タブで、ステップ 2 で S3 バケットにアップロードしたファイルと同じ名前のファイルを選択します(例、`tutorial.txt`)。

   このファイルには、変換されたすべてのデータが含まれているはずです。
5. 変換されたデータを表示するには、[開く] または [ダウンロード] を選択します。

Python スクリプトを実行して、元のデータと変換されたデータを出力する

S3 Object Lambda は、既存のアプリケーションで使用できます。これを行うには、アプリケーション設定を更新して、ステップ 6 で、S3 からのデータを作成した新しい S3 Object Lambda アクセスポイント ARN を使用してデータを取得します。

次の Python スクリプトの例では、S3 バケットからの元のデータと S3 Object Lambda アクセスポイントからの変換されたデータの両方を出力します。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Object Lambda アクセスポイント] を選択します。
3. [Object Lambda アクセスポイント] ページで、ステップ 6 で作成した S3 Object Lambda アクセスポイントの左にあるラジオボタンを選択します (例、**`tutorial-object-lambda-accesspoint`**)。
4. [ARN のコピー] を選択します。
5. 後で使用するために ARN を保存します。

6. ローカルマシンで Python スクリプトを記述して、S3 バケットからの元のデータ (例えば、`tutorial.txt`) と S3 Object Lambda アクセスポイントからの変換されたデータ (例えば、`tutorial.txt`) の両方を出力します。次のサンプルスクリプトを使用できます。

```
import boto3
from botocore.config import Config

s3 = boto3.client('s3', config=Config(signature_version='s3v4'))

def getObject(bucket, key):
    objectBody = s3.get_object(Bucket = bucket, Key = key)
    print(objectBody["Body"].read().decode("utf-8"))
    print("\n")

print('Original object from the S3 bucket:')
# Replace the two input parameters of getObject() below with
# the S3 bucket name that you created in Step 1 and
# the name of the file that you uploaded to the S3 bucket in Step 2
getObject("tutorial-bucket",
          "tutorial.txt")

print('Object transformed by S3 Object Lambda:')
# Replace the two input parameters of getObject() below with
# the ARN of your S3 Object Lambda Access Point that you saved earlier and
# the name of the file with the transformed data (which in this case is
# the same as the name of the file that you uploaded to the S3 bucket
# in Step 2)
getObject("arn:aws:s3-object-lambda:us-west-2:111122223333:accesspoint/tutorial-
object-lambda-accesspoint",
          "tutorial.txt")
```

7. Python スクリプトにカスタム名 (`tutorial_print.py` など) を付けて、ステップ 4 でローカルマシンに作成したフォルダ (`object-lambda` など) に保存します。

8. ローカルターミナルで、ステップ 4 で作成したディレクトリ (例えば、`object-lambda`) のルートから次のコマンドを実行します。

```
python3 tutorial_print.py
```

元のデータと変換されたデータ(すべて大文字)の両方がターミナルに表示されます。例えば、次のようなテキストが表示されます。

```
Original object from the S3 bucket:
Amazon S3 Object Lambda Tutorial:
You can add your own code to process data retrieved from S3 before
returning it to an application.

Object transformed by S3 Object Lambda:
AMAZON S3 OBJECT LAMBDA TUTORIAL:
YOU CAN ADD YOUR OWN CODE TO PROCESS DATA RETRIEVED FROM S3 BEFORE
RETURNING IT TO AN APPLICATION.
```

ステップ 8: クリーンアップする

学習のためだけに S3 Object Lambda によってデータを変換した場合は、割り当てた AWS リソースを削除して、料金が発生しないようにします。

サブステップ

- Object Lambda アクセスポイントの削除
- S3 アクセスポイントを削除する
- Lambda 関数の実行ロールを削除する
- Lambda 関数を削除する
- CloudWatch Logs グループを削除する
- S3 ソースバケットの元のファイルを削除する
- S3 ソースバケットを削除する
- IAM ロールを削除する

Object Lambda アクセスポイントの削除

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Object Lambda アクセスポイント] を選択します。
3. [Object Lambda アクセスポイント] ページで、ステップ 6 で作成した S3 Object Lambda アクセスポイントの左にあるラジオボタンを選択します (例、**`tutorial-object-lambda-accesspoint`**)。
4. [削除] を選択します。

5. 表示されるテキストフィールドにアクセスポイントの名前を入力して、[削除] を選択し、bject Lambda アクセスポイントを削除することを確認します。

S3 アクセスポイントを削除する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[アクセスポイント] を選択します。
3. ステップ 3 で作成したアクセスポイントに移動し (例、**tutorial-access-point**)、アクセスポイントの名前の横にあるラジオボタンを選択します。
4. [削除] を選択します。
5. 表示されるテキストフィールドにアクセスポイントの名前を入力して、アクセスポイントを削除することを確認し、[削除] を選択します。

Lambda 関数の実行ロールを削除する

1. AWS Management Console にサインインして AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. 左側のナビゲーションペインで、[関数] を選択します。
3. ステップ 4 で作成した関数 (例、**tutorial-object-lambda-function**) を選択します。
4. Lambda 関数の詳細ページで、[設定] タブを選択してから、左のナビゲーションペインの [許可] を選択します。
5. [実行ロール] で、[ロール名] のリンクを選択します。IAM コンソールが開きます。
6. Lambda 関数の実行ロールの IAM コンソールの [概要] ページで、[ロールの削除] を選択します。
7. ロールの削除ダイアログボックスで、[はい、削除します] をクリックします。

Lambda 関数を削除する

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) で、左のナビゲーションペインの関数を選択します。
2. ステップ 4 で作成した関数名の左にあるチェックボックスを選択します (例、**tutorial-object-lambda-function**)。

3. [アクション] を選択し、[削除] を選択します。
4. [関数の削除] ダイアログボックスで、[削除] を選択します。

CloudWatch Logs グループを削除する

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 左側のナビゲーションペインで、[ロググループ] をクリックします。
3. ステップ 4 で作成した Lambda 関数で終わる名前のロググループを検索します (例、**tutorial-object-lambda-function**)。
4. ロググループの名前の左にあるチェックボックスを選択します。
5. [アクション] を選択してから、[ロググループの削除] を選択します。
6. [ロググループの削除] ダイアログボックスで、[削除] をクリックします。

S3 ソースバケットの元のファイルを削除する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. バケット名] リストで、ステップ 2 で元のファイルをアップロードしたバケットの名前を選択します(例、**tutorial-bucket**)。
4. 削除するオブジェクトの名前の左にあるチェックボックスを選択します (例、`tutorial.txt`)。
5. [削除] を選択します。
6. [オブジェクトの削除] ページの [オブジェクトを完全に削除しますか?] セクションで、テキストボックスに「**permanently delete**」と入力して、このオブジェクトを削除することを確認します。
7. [オブジェクトの削除] を選択します。

S3 ソースバケットを削除する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。

3. [バケット] リストで、ステップ 1 で作成したバケットの名前の横にあるラジオボタンを選択します (例、**tutorial-bucket**)。
4. [削除] を選択します。
5. [バケットを削除する] ページで、テキストフィールドにバケット名を入力することでバケットを削除することを確認し、[バケットを削除する] を選択します。

IAM ロールを削除する

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ユーザー] を選択し、削除するユーザー名の横にあるチェックボックスを選択します。
3. ページの上部で、[削除] を選択します。
4. [*****を削除しますか?] ダイアログボックスで、テキスト入力フィールドにユーザー名を入力し、ユーザーの削除を確認します。[削除] を選択します。

次のステップ

このチュートリアルを完了すると、ユースケースの Lambda 関数をカスタマイズし、標準 S3 GET リクエストによって返されるデータを変更できます。

S3 Object Lambda の一般的なユースケースを以下に示します。

- セキュリティとコンプライアンスのために機密データをマスキングします。

  詳細については、「チュートリアル: S3 Object Lambda と Amazon Comprehend を使用した PII データの検出と編集」を参照してください。
- 特定の情報を配信するために、特定のデータ行をフィルタリングします。
- 他のサービスやデータベースからの情報でデータ強化します。
- アプリケーション互換性のために XML から JSON への変換など、データ形式間の変換を行います。
- ダウンロード中のファイルを圧縮または解凍します。
- 画像のサイズ変更とウォーターマークを行います。

  詳細については、「チュートリアル: S3 Object Lambda を使用して、取得時に画像に動的に透かしを入れる」を参照してください。

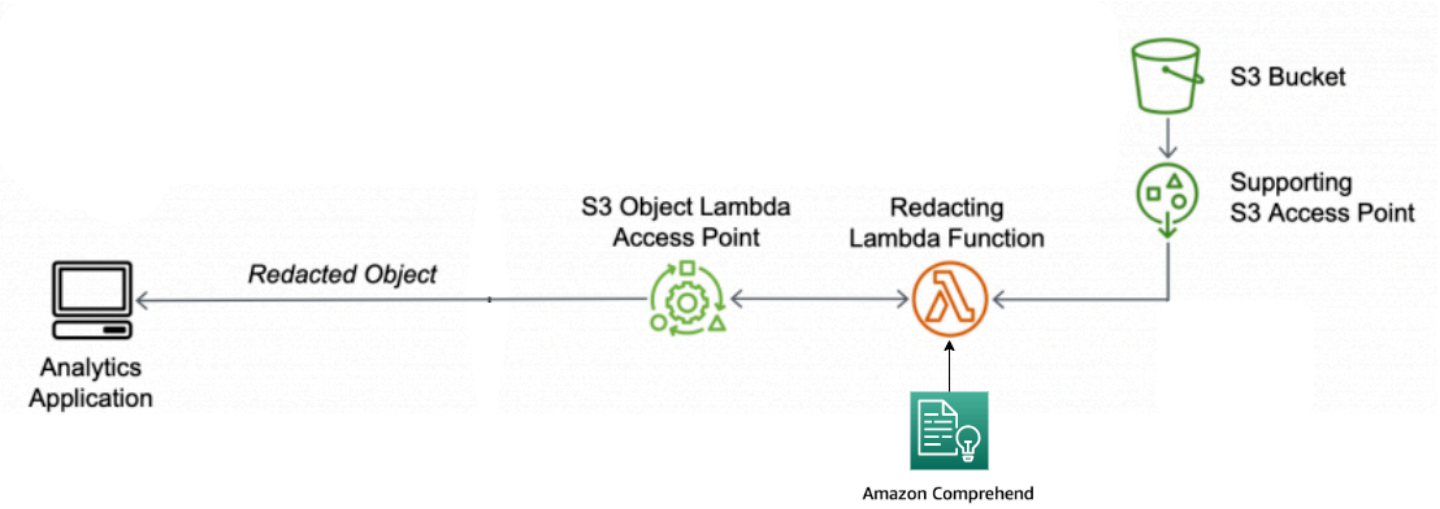
- データにアクセスするためのカスタム承認規則を実装します。

S3 Object Lambda, の詳細については、「S3 Object Lambda を使用したオブジェクトの変換」を参照してください。

チュートリアル: S3 Object Lambda と Amazon Comprehend を使用した PII データの検出と編集

複数のアプリケーションおよびユーザーがアクセスするための共有データセットに Amazon S3 を使用する場合、個人を特定できる情報 (PII) などの特権情報を許可されたエンティティのみに制限することが重要です。例えば、マーケティングアプリケーションが PII を含むデータを使用する場合、データのプライバシー要件を満たすために、まず PII データをマスクする必要があります。また、分析アプリケーションで製造オーダーのインベントリデータセットを使用する場合、意図しないデータ漏洩を防ぐために、まず顧客のクレジットカード情報を編集する必要があります。

S3 Object Lambda と Amazon Comprehend により事前に構築された AWS Lambda 関数を使用すると、S3 から取得した PII データを、アプリケーションに返す前に保護できます。具体的な内容は以下のとおりです。組み込みの Lambda 関数を編集関数として作使用し、S3 Object Lambda アクセスポイントにアタッチします。アプリケーション (Analytics アプリケーションなど) が標準 S3 GET リクエストを送信する場合、S3 Object Lambda アクセスポイントを介して行われたこれらのリクエストは、事前作成された編集用 Lambda 関数を呼び出し、サポートしている S3 アクセスポイントを介して S3 バケットから取得した PII データを検出して修正します。次に、S3 Object Lambda アクセスポイントは、編集された結果をアプリケーションに返します。



プロセスでは、事前に構築された Lambda 関数は Amazon Comprehend (自然言語処理 (NLP) サービス) を使用して、PII がテキストにどのように存在するかに関係なく (数値、または単語と数字の組

み合わせなど)、PII の表現方法のバリエーションをキャプチャします。Amazon Comprehend は、テキスト内のコンテキストを使用して、4 桁の数字が PIN、社会保障番号 (SSN) の最後の 4 桁の数字であるか、または年であるかを理解することもできます。Amazon Comprehend は UTF-8 形式のテキストファイルを処理し、精度に影響を与えずに PII を大規模に保護します。詳細については、Amazon Comprehend デベロッパーガイドの「Amazon Comprehend とは?」を参照してください。

目的

このチュートリアルでは、事前に構築された Lambda 関数 `ComprehendPiiRedactionS3ObjectLambda` で S3 Object Lambda を使用する方法を学習します。この関数は、Amazon Comprehend を使用して PII エンティティを検出します。次に、これらのエンティティをアスタリスクに置き換えて墨消しします。PII を編集することで、機密データを隠すことができ、セキュリティとコンプライアンスに役立ちます。

また、AWS Serverless Application Repository で事前作成された AWS Lambda 関数を操作して、S3 Object Lambda と連携してデプロイメントを容易にする方法も学びます。

トピック

前提条件: 許可を持つ IAM ユーザーを作成する

このチュートリアルを開始する前に、正しいアクセス許可を持つ AWS Identity and Access Management ユーザー (IAM ユーザー) としてサインインできる AWS アカウントが必要です。

チュートリアル用の IAM ユーザーを作成できます。このチュートリアルを完了するには、IAM ユーザーが次の IAM ポリシーをアタッチして、関連する AWS リソースにアクセスし、特定のアクションを実行する必要があります。

> **Note**
>
> このチュートリアルでは、わかりやすいように IAM ユーザーを作成して使用します。このチュートリアルを完了したら、忘れずに「IAM ユーザーを削除する」を行います。本番環境で使用する場合は、IAM ユーザーガイドの「IAM でのセキュリティのベストプラクティス」に従うことをお勧めします。ベストプラクティスでは、人間のユーザーと ID プロバイダーとのフェデレーションにより、一時的な認証情報を使用して AWS にアクセスする必要があります。追加のベストプラクティスとして、ワークロードでは一時的な認証情報で IAM ロールを使用し、AWS にアクセス必要があります。AWS IAM Identity Center を使用して一時的な認証情報を持つユーザーを作成する方法については、「AWS IAM Identity Center ユーザーガイド」の「開始方法」を参照してください。
> このチュートリアルでは、フルアクセスポリシーも使用します。実稼働環境では、代わりに、セキュリティのベストプラクティスに従って、ユースケースに必要な最小限の許可のみを付与することをお勧めします。

IAM ユーザーには次のものが必要です。AWS マネージドポリシー:

- AmazonS3FullAccess – Object Lambda アクセスポイントを作成および使用するための許可を含む、すべての Amazon S3 アクションに対する許可を付与します。
- AWSLambda_FullAccess – すべての Lambda アクションに許可を付与します。
- AWSCloudFormationFullAccess – すべてのAWS CloudFormationのアクションに許可を付与します。
- IAMFullAccess – すべての IAM アクションに許可を付与します。
- IAMAccessAnalyzerReadOnlyAccess – IAM Access Analyzer によって提供されるすべてのアクセス情報を読み取る許可を付与します。

IAM ユーザーを作成するときに、これらの既存のポリシーを直接アタッチできます。IAM ユーザーを作成する方法の詳細については、「IAM ユーザーガイド」の「IAM ユーザーの作成 (コンソール)」を参照してください。

さらに、IAM ユーザーにはカスタマーマネージドポリシーが必要です。IAM ユーザーにすべての AWS Serverless Application Repository リソースとアクションへの許可を付与するには、IAM ポリシーを作成し、ポリシーを IAM ユーザーにアタッチする必要があります。

IAM ポリシーを作成して IAM ユーザーにアタッチするには

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインの [ポリシー] を選択します。
3. [Create policy] を選択します。
4. Service の [Visual editor] タブで、[サービスの選択] を選択します。次に、[Serverless Application Repository] を選択します。
5. [アクション] の [手動アクション] の下で、このチュートリアルでは、[すべての Serverless Application Repository アクション (serverlessrepo:*)] を選択します。

   セキュリティのベストプラクティスとは、ユーザーが必要とするアクションとリソースに対する許可を、ユースケースに基づいて付与することです。詳細については、IAM ユーザーガイドの「IAM でのセキュリティベストプラクティス」を参照してください。
6. [リソース] で、このチュートリアルでは、[すべてのリソース] を選択します。

   ベストプラクティスは、特定のアカウントの特定のリソースに対してのみ許可を定義することです。または、条件キーを使用して、最小の特権を付与することもできます。詳細については、IAM ユーザーガイドの「最小限の特権を付与」を参照してください。
7. [Next: Tags] (次へ: タグ) を選択します。
8. [次へ: レビュー] を選択します。
9. [ポリシーの確認] ページで、作成するポリシーの [名前] (例、**`tutorial-serverless-application-repository`**) と [説明] (オプション) を入力します。ポリシー概要を確認して、目的のアクセス許可を付与していることを確認し、[Create policy] (ポリシーの作成) を選択して新しいポリシーを保存します。
10. 左のナビゲーションペインで、[ユーザー] を選択します。次に、このチュートリアルの IAM ユーザーを選択します。
11. 選択したユーザーの [概要] ページで、[許可] タブを選択してから、[許可を追加] を選択します。
12. [権限の付与] で、[既存のポリシーを直接アタッチします] を選択します。
13. 今、作成したポリシーの横にあるチェックボックスを選択し (例、**`tutorial-serverless-application-repository`**)、次に [次へ: レビュー] を選択します。
14. [許可概要] で、概要を確認し、意図したポリシーを付与したことを確認します。次に、[許可を追加] を選択します。

ステップ 1: S3 バケットを作成する。

変換する元のデータを保存するバケットを作成します。

バケットを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [Create bucket (バケットの作成)] を選択します。

   [バケットの作成] ページが開きます。
4. [バケット名] に、バケットの名前 (例、**tutorial-bucket**) を入力します。

   Amazon S3 のバケット命名規則の詳細については、「バケットの名前付け」を参照してください。
5. [リージョン] で、バケットを保存する AWS リージョンを選択します。

   バケットのリージョンの詳細については、「バケットの概要」を参照してください。
6. [このバケットのパブリックアクセス設定をブロック] で、デフォルト設定 (ブロックすべてパブリックアクセスが有効) のままであることを確認します。

   ユースケースでオフにする必要のある設定が 1 つ以上あることがわかっている場合を除き、すべてのブロックパブリックアクセス設定を有効にしておくことをお勧めします。ブロックパブリックアクセスの詳細については、Amazon S3 ストレージへのパブリックアクセスのブロックを参照してください。
7. 残りの設定はデフォルトのままにしておきます。

   (オプション) 特定のユースケースに追加のバケット設定を設定する場合は、「バケットの作成」を参照してください。
8. [Create bucket (バケットの作成)] を選択します。

ステップ 2: S3 バケットにファイルをアップロード

名前、銀行情報、電話番号、SSN など、さまざまなタイプの既知の PII データを含むテキストファイルを S3 バケットにアップロードします。このテキストファイルは、このチュートリアルの後半で PII を編集します。

例えば、次の tutorial.txt ファイルをアップロードできます。これは、Amazon Comprehend からの入力ファイルの例です。

```
Hello Zhang Wei, I am John. Your AnyCompany Financial Services,
LLC credit card account 1111-0000-1111-0008 has a minimum payment
of $24.53 that is due by July 31st. Based on your autopay settings,
we will withdraw your payment on the due date from your
bank account number XXXXXX1111 with the routing number XXXXX0000.

Your latest statement was mailed to 100 Main Street, Any City,
WA 98121.
After your payment is received, you will receive a confirmation
text message at 206-555-0100.
If you have questions about your bill, AnyCompany Customer Service
is available by phone at 206-555-0199 or
email at support@anycompany.com.
```

バケットにファイルをアップロードするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、ステップ 1 で作成したバケットの名前を選択し (**tutorial-bucket** など)、ファイルをアップロードします。
4. バケットの [オブジェクト] タブで、[アップロード] を選択します。
5. [アップロード] ページの [ファイルとフォルダ] の下で、[ファイルを追加] を選択します。
6. アップロードするファイルを選択し、続いて [オープン] を選択します。例えば、前述した tutorial.txt ファイルの例をアップロードできます。
7. [アップロード] を選択します。

ステップ 3: S3 アクセスポイントの作成

S3 Object Lambda アクセスポイントを使用して元のデータにアクセスし、変換するには、S3 アクセスポイントを作成し、ステップ 1 で作成した S3 バケットに関連付けます。アクセスポイントは、変換するオブジェクトと同じ AWS リージョン に存在する必要があります。

このチュートリアルの後半では、このアクセスポイントを Object Lambda アクセスポイントのサポートアクセスポイントとして使用します。

アクセスポイントを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[アクセスポイント] を選択します。
3. [アクセスポイント] ページで、[アクセスポイントの作成] を選択します。
4. [アクセスポイント名] フィールドに、アクセスポイントの名前を入力します (**tutorial-pii-access-point** など)。

   アクセスポイントの名前付けの詳細については、「Amazon S3 アクセスポイントの命名規則」を参照してください。
5. [バケット名] フィールドに、ステップ 1 で作成したバケットの名前を入力します(**tutorial-bucket** など)。S3 は、アクセスポイントをこのバケットにアタッチします。

   (オプション) [S3 を参照] を選択して、アカウント内のバケットを参照および検索できます。[S3 を参照] を選択した場合は、目的のバケットを選択し、[パスの選択] を選択して、[バケット名] フィールドにバケットの名前を入力します。
6. [ネットワークオリジン] で、[インターネット] を選択します。

   アクセスポイントのネットワークオリジンの詳細については、「Virtual Private Cloud に制限されたアクセスポイントの作成」を参照してください。
7. デフォルトでは、アクセスポイントに対してすべてのブロックパブリックアクセス設定が有効になります。パブリックアクセスブロック設定は、すべてを有効にしておくことを推奨します。詳細については、「アクセスポイントへのパブリックアクセスの管理」を参照してください。
8. その他のすべてのアクセスポイント設定で、デフォルトの設定を保持します。

   (オプション) アクセスポイントの設定は、ユースケースをサポートするように変更できます。このチュートリアルでは、デフォルトの設定を保持することをお勧めします。

   (オプション) アクセスポイントへのアクセスを管理する必要がある場合は、アクセスポイントポリシーを指定できます。詳細については、「アクセスポイントポリシーの例」を参照してください。
9. [アクセスポイントを作成] を選択します。

ステップ 4: 事前構築された Lambda 関数の設定とデプロイ

PII データを編集するには、事前に構築された AWS Lambda関数 `ComprehendPiiRedactionS3ObjectLambda` を S3 Object Lambda アクセスポイントと共に設定、デプロイします。

Lambda 関数の設定とデプロイ

1. AWS Management Console にサインインし、AWS Serverless Application Repository で `ComprehendPiiRedactionS3ObjectLambda` 関数を表示します。
2. [アプリケーション名] の下の [アプリケーションの設定] では、このチュートリアルのデフォルト値 (`ComprehendPiiRedactionS3ObjectLambda`) を保持します。

   (オプション) このアプリケーションに付ける名前を入力できます。同じ共有データセットに対して異なるアクセスニーズを持つ複数の Lambda 関数を設定する場合は、これを行うことをお勧めします。
3. [MaskCharacter] で、デフォルト値 (*) を保持します。マスク文字は、編集された PII エンティティの各文字を置き換えます。
4. [MaskMode] で、デフォルト値 (MASK) を保持します。MaskMode の値は、PII エンティティを `MASK` 文字または `PII_ENTITY_TYPE` 値に変更することを指定します。
5. 指定したタイプのデータを墨消しするには、[PiiEntityTypes] で、デフォルト値の ALL を保持します。PiiEntityTypes の値は、墨消しの対象となる PII エンティティタイプを指定します。

   サポートされる PII エンティティタイプの一覧の詳細については、Amazon Comprehend デベロッパーガイドの個人を特定できる情報 (PII) の検出を参照してください。
6. 残りの設定はデフォルト値のままにしておきます。

   (オプション) 特定のユースケースに対して追加の設定を構成する場合は、ページの左側の Readme ファイルセクションを参照してください。
7. [I acknowledge that this app creates custom IAM roles (このアプリでカスタム IAM ロールを作成することを認識しています)] チェックボックスをオンにします。
8. [デプロイ] を選択します。
9. 新しいアプリケーションのページの [リソース] で、デプロイする Lambda 関数の [論理 ID] を選択して、Lambda 関数ページで関数を確認します。

ステップ 5: S3 Object Lambda アクセスポイントの作成

S3 Object Lambda アクセスポイントは、S3 GET リクエストから Lambda 関数を直接呼び出す柔軟性を提供し、関数が S3 アクセスポイントから取得した PII データを編集できるようにします。S3 Object Lambda アクセスポイントを作成および設定するときは、変更する Lambda 関数を指定して、Lambda が使用するカスタムパラメータとして JSON 形式のイベントコンテキストを呼び出し、提供する必要があります。

イベントコンテキストは、S3 Object Lambda から Lambda に渡されたイベントで行われたリクエストに関する情報を提供します。イベントコンテキスト内のすべてのフィールドの詳細については、「イベントコンテキストの形式と使用法」を参照してください。

S3 Object Lambda アクセスポイントを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Object Lambda アクセスポイント] を選択します。
3. [Object Lambda アクセスポイント] ページで、[Object Lambda アクセスポイントの作成] を選択します。
4. Object Lambda アクセスポイント名] に、Object Lambda アクセスポイントに使用する名前を入力します (例、**`tutorial-pii-object-lambda-accesspoint`**)。
5. [サポートするアクセスポイント] で、ステップ 3 で作成した標準アクセスポイントを入力または参照します (**`tutorial-pii-access-point`** など)。次に、[サポートするアクセスポイントの選択] を選択します。
6. S3 API の場合、Lambda 関数が処理する S3 バケットからオブジェクトを取得するには、[GetObject] を選択します。
7. [Lambda 関数の呼び出し] では、このチュートリアルでは、次の 2 つのオプションのいずれかを選択できます。
   - [アカウントの関数から選ぶ] を選択し、[Lambda 関数] ドロップダウンリストから、ステップ 4 でデプロイした Lambda 関数を選択します (例、**`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**)。
   - ARN を入力] を選択し、ステップ 4 で作成した Lambda 関数の Amazon リソースネーム (ARN) を入力します。
8. [Lambda 関数のバージョン] で、$LATEST (ステップ 4 でデプロイした Lambda 関数の最新バージョン) を選択します。

9. (オプション) Lambda 関数で、範囲とパート番号によって GET リクエストを認識および処理する必要がある場合は、[Lambda 関数は、範囲を使用してリクエストをサポート] および [Lambda 関数は、パート番号を使用してリクエストをサポート] を選択します。それ以外の場合は、これらの 2 つのチェックボックスをオフにします。

   S3 Object Lambda で範囲またはパート番号を使用する方法の詳細については、「Range および partNumber ヘッダーの操作」を参照してください。

10. (オプション) [ペイロード - オプション] で JSON テキストを追加して、Lambda 関数に追加情報を提供します。

    ペイロードは、特定の S3 Object Lambda アクセスポイントからのすべての呼び出しに対する入力として、Lambda 関数に提供できるオプションの JSON テキストです。同じ Lambda 関数を呼び出す複数の Object Lambda アクセスポイントの動作をカスタマイズして、異なるパラメータを使用してペイロードを設定できます。これにより、Lambda 関数の柔軟性が向上します。

    ペイロードの詳細については、「イベントコンテキストの形式と使用法」を参照してください。

11. (オプション) [リクエストメトリクス - オプション] で、[無効] または [有効] を選択して、Amazon S3 モニタリングを Object Lambda アクセスポイントに追加します。リクエストメトリクスには、Amazon CloudWatch の標準料金が課金されます。詳細については、「CloudWatch 料金表」を参照してください。

12. [Object Lambda アクセスポイントポリシー-オプション] で、デフォルトの設定を保持します。

    (オプション) リソースポリシーを設定できます。このリソースポリシーは、指定された Object Lambda アクセスポイントを使用する `GetObject` API 許可を付与します。

13. 残りの設定はデフォルト値のままにしておき、[Object Lambda アクセスポイントの作成] を選択します。

ステップ 6: S3 Object Lambda アクセスポイントを使用して、編集されたファイルを取得する

これで、S3 Object Lambda は元のファイルから PII データを編集する準備が整いました。

S3 Object Lambda アクセスポイントを使用して、編集されたファイルを取得するには

S3 Object Lambda アクセスポイントを使用してファイルの取得をリクエストすると、S3 Object Lambda への `GetObject` API 呼び出しが行われます。S3 Object Lambda は Lambda 関数を呼び出して PII データを変換し、変換されたデータを標準 S3 `GetObject` API への応答として返します。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Object Lambda アクセスポイント] を選択します。
3. [Object Lambda アクセスポイント] ページで、ステップ 5 で作成した S3 Object Lambda アクセスポイントを選択します (例、**`tutorial-pii-object-lambda-accesspoint`**)。
4. S3 Object Lambda アクセスポイントの [オブジェクト] タブで、ステップ 2 で S3 バケットにアップロードしたファイルと同じ名前のファイルを選択します(例、`tutorial.txt`)。

   このファイルには、変換されたすべてのデータが含まれているはずです。
5. 変換されたデータを表示するには、[開く] または [ダウンロード] を選択します。

   次の例に示すように、編集されたファイルを確認できます。

```
Hello *********. Your AnyCompany Financial Services,
LLC credit card account ****************** has a minimum payment
of $24.53 that is due by *********. Based on your autopay settings,
we will withdraw your payment on the due date from your
bank account ********** with the routing number *********.

Your latest statement was mailed to **********************************.
After your payment is received, you will receive a confirmation
text message at ************.
If you have questions about your bill, AnyCompany Customer Service
is available by phone at ************ or
email at **********************.
```

ステップ 7: クリーンアップする

学習のためだけに S3 Object Lambda によってデータを編集した場合は、割り当てた AWS リソースを削除して、料金が発生しないようにします。

サブステップ

- Object Lambda アクセスポイントの削除
- S3 アクセスポイントを削除する
- Lambda 関数を削除する
- CloudWatch Logs グループを削除する

- S3 ソースバケットの元のファイルを削除する
- S3 ソースバケットを削除する
- Lambda 関数の IAM ロールを削除する
- IAM ユーザーのカスタマーマネージドポリシーを削除する
- IAM ユーザーを削除する

Object Lambda アクセスポイントの削除

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Object Lambda アクセスポイント] を選択します。
3. [Object Lambda アクセスポイント] ページで、ステップ 5 で作成した S3 Object Lambda アクセスポイントの左にあるオプションブタンを選択します (例、**tutorial-pii-object-lambda-accesspoint**)。
4. [削除] を選択します。
5. 表示されるテキストフィールドにアクセスポイントの名前を入力して、[削除] を選択し、bject Lambda アクセスポイントを削除することを確認します。

S3 アクセスポイントを削除する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[アクセスポイント] を選択します。
3. ステップ 3 で作成したアクセスポイントに移動し (例、**tutorial-pii-access-point**)、アクセスポイントの名前の横にあるオプションボタンを選択します。
4. [削除] を選択します。
5. 表示されるテキストフィールドにアクセスポイントの名前を入力して、アクセスポイントを削除することを確認し、[削除] を選択します。

Lambda 関数を削除する

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) で、左のナビゲーションペインの関数を選択します。

2. ステップ 4 で作成した関数 (例、**serverlessrepo-ComprehendPiiRedactionS3ObjectLambda**) を選択します。
3. [アクション] を選択し、[削除] を選択します。
4. [関数の削除] ダイアログボックスで、[削除] を選択します。

CloudWatch Logs グループを削除する

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 左側のナビゲーションペインで、[ロググループ] をクリックします。
3. ステップ 4 で作成した Lambda 関数で終わる名前のロググループを検索します (例、**serverlessrepo-ComprehendPiiRedactionS3ObjectLambda**)。
4. [アクション] を選択してから、[ロググループの削除] を選択します。
5. [ロググループの削除] ダイアログボックスで、[削除] をクリックします。

S3 ソースバケットの元のファイルを削除する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. バケット名] リストで、ステップ 2 で元のファイルをアップロードしたバケットの名前を選択します(例、**tutorial-bucket**)。
4. 削除するオブジェクトの名前の左にあるチェックボックスを選択します (例、`tutorial.txt`)。
5. [削除] を選択します。
6. [オブジェクトの削除] ページの [オブジェクトを完全に削除しますか?] セクションで、テキストボックスに「**permanently delete**」と入力して、このオブジェクトを削除することを確認します。
7. [オブジェクトの削除] を選択します。

S3 ソースバケットを削除する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、ステップ 1 で作成したバケットの名前の横にあるラジオボタンを選択します (例、**tutorial-bucket**)。
4. [削除] を選択します。
5. [バケットを削除する] ページで、テキストフィールドにバケット名を入力することでバケットを削除することを確認し、[バケットを削除する] を選択します。

Lambda 関数の IAM ロールを削除する

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択し、削除するロール名の隣にあるチェックボックスを選択します。ロール名は、ステップ 4 でデプロイした Lambda 関数の名前で始まります (例、**serverlessrepo-ComprehendPiiRedactionS3ObjectLambda**)。
3. [削除] を選択します。
4. [削除] ダイアログボックスで、テキスト入力フィールドにロール名を入力し、ユーザーの削除を確認します。その後、[Delete] (削除) をクリックします。

IAM ユーザーのカスタマーマネージドポリシーを削除する

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインの [ポリシー] を選択します。
3. [ポリシー] ページで、[前提条件] で作成したカスタマーマネージドポリシーの名前 (例、**tutorial-serverless-application-repository**) を検索ボックスに入力して、ポリシーのリストをフィルタリングします。削除するアポリシーの名前の横にあるオプションボタンを選択します。
4. [アクション] を選択し、[削除] を選択します。
5. 表示されるテキストフィールドにポリシーの名前を入力して、このポリシーを削除することを確認し、[削除] を選択します。

IAM ユーザーを削除する

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ユーザー] を選択し、削除するユーザー名の横にあるチェックボックスを選択します。
3. ページの上部で、[削除] を選択します。
4. [*****を削除しますか?] ダイアログボックスで、テキスト入力フィールドにユーザー名を入力し、ユーザーの削除を確認します。[削除] を選択します。

次のステップ

このチュートリアルを完了すると、次の関連するユースケースをさらに調べることができます。

- 複数の S3 Object Lambda アクセスポイントを作成し、データアクセサーのビジネスニーズに応じて特定の種類の PII を変更するように構成された事前構築済みの Lambda 関数でそれらを有効にすることができます。

  各タイプのユーザーは IAM ロールを引き受け、1 つの S3 Object Lambda アクセスポイント(IAM ポリシーで管理)にのみアクセスできます。次に、異なる編集ユースケース用に設定されたそれぞれの `ComprehendPiiRedactionS3ObjectLambda` Lambda 関数を異なる S3 Object Lambda アクセスポイントにアタッチします。S3 Object Lambda アクセスポイントごとに、共有データセットを保存する S3 バケットからデータを読み取るためのサポートする S3 アクセスポイントを持つことができます。

  ユーザーが、S3 アクセスポイントを介してのみバケットから読み取ることを許可する、S3 バケットポリシーを作成する方法については、「アクセスポイントを使用するための IAM ポリシーの設定」を参照してください。

  Lambda 関数、S3 アクセスポイント、および S3 Object Lambda アクセスポイントにアクセスする許可をユーザーに付与する方法の詳細については、「Object Lambda アクセスポイントの IAM ポリシーの設定」を参照してください。
- 独自の Lambda 関数を構築し、カスタマイズした Lambda 関数で S3 Object Lambda を使用し、特定のデータニーズを満たすことができます。

  例えば、さまざまなデータ値を調べるには、S3 Object Lambda と、追加の Amazon Comprehend の機能(エンティティ認識、キーフレーズ認識、センチメント分析、ドキュメントの分類など)を

使用して、データを処理できます。また、S3 Object Lambda を Amazon Comprehend Medical (HIPAA 適格な NLP サービス) と共に使用して、コンテキストに応じた方法でデータを分析および抽出できます。

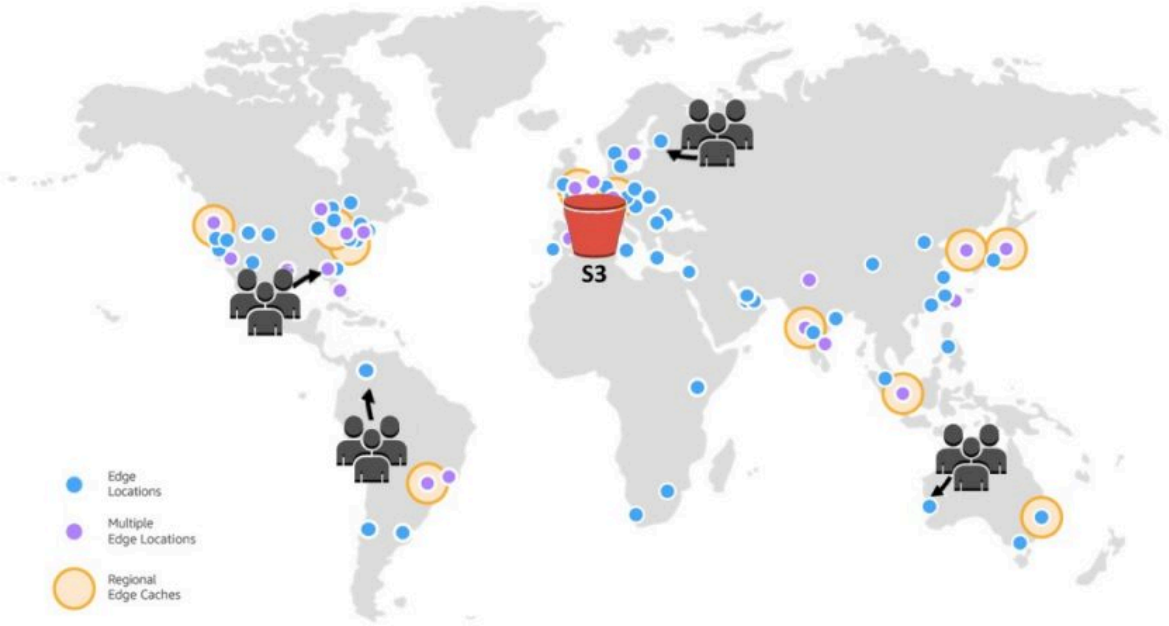
S3 Object Lambda および独自の Lambda 関数を使用してデータを変換する方法の詳細については、「チュートリアル: S3 Object Lambda を使用したアプリケーションのデータの変換」を参照してください。

チュートリアル: Amazon S3、Amazon CloudFront、Amazon Route 53 を使用したオンデマンドストリーミング動画のホスティング。

Amazon S3 を Amazon CloudFront とともに使用すると、セキュリティとスケーラビリティを備えたオンデマンド視聴用に動画をホストできます。オンデマンド動画 (VOD) ストリーミングの場合、動画コンテンツはサーバーに保存され、ビューワーはいつでも視聴できます。

CloudFront は、高速で安全性が高く、プログラム可能なコンテンツ配信ネットワーク (CDN) サービスです。CloudFront は、CloudFront のすべてのエッジロケーションから HTTPS 経由でコンテンツを安全に配信できます。CloudFront の詳細については、「Amazon CloudFront デベロッパーガイド」の「Amazon CloudFront とは」を参照してください。

CloudFront のキャッシングは、オリジンサーバーが直接応答するリクエストの数を減らします。ビューワー (エンドユーザー) が CloudFront で配信する動画をリクエストすると、リクエストはビューワーの場所により近いエッジロケーションにルーティングされます。CloudFront はキャッシュから動画を提供し、まだキャッシュされていない場合にのみ、S3 バケットから動画を取得します。管理機能のキャッシュにより、低レイテンシー、高スループット、高速転送速度で、世界中のビューワーに動画を配信できます。CloudFront キャッシュ管理の詳細については、Amazon CloudFront デベロッパーガイドのキャッシュの最適化と可用性を参照してください。



目的

このチュートリアルでは、配信に CloudFront を使用し、ドメインネームシステム (DNS) およびカスタムドメイン管理に Amazon Route 53 を使用して、オンデマンド動画ストリーミングをホストするように S3 バケットを設定します。

トピック

- 前提条件: カスタムドメインを Route 53 に登録し、設定する。
- ステップ 1: S3 バケットを作成する。
- ステップ 2: S3 バケットに動画をアップロードする。
- ステップ 3: CloudFront オリジンアクセスアイデンティティを作成する。
- ステップ 4: CloudFront ディストリビューションを作成する。
- ステップ 5: CloudFront ディストリビューション経由で動画にアクセスする。
- ステップ 6: カスタムドメイン名を使用するように CloudFront ディストリビューションを設定する。
- ステップ 7: カスタムドメイン名を使用して CloudFront ディストリビューションを介して S3 動画にアクセスする。

前提条件: カスタムドメインを Route 53 に登録し、設定する。

このチュートリアルを開始する前に、Route 53 により、後でカスタムドメイン名を使用するように CloudFront ディストリビューションを設定するために、カスタムドメイン (例、**example.com**) を登録し、設定する必要があります 。

カスタムドメイン名がなければ、S3 動画はパブリックにアクセスでき、CloudFront 経由でホストされる URL は、次のようになります。

```
https://CloudFront distribution domain name/Path to an S3 video
```

例えば、**https://d111111abcdef8.cloudfront.net/sample.mp4** と指定します。

Route 53 で設定されたカスタムドメイン名を使用するように CloudFront ディストリビューションを設定すると、S3 ビデオはパブリックにアクセス可能になり、CloudFront を介して次のような URL でホストされます。

```
https://CloudFront distribution alternate domain name/Path to an S3 video
```

例えば、**https://www.example.com/sample.mp4** と指定します。カスタムドメイン名は、ビューワーが使用するのに簡単で直感的です。

カスタムドメインの登録については、Amazon Route 53 デベロッパーガイド の「Route 53 を使用して新しいドメインを登録する」を参照してください。

Route 53 にドメイン名を登録すると、Route 53 によってホストゾーンが作成されます。このホストゾーンは、このチュートリアルの後半で使用します。このホストゾーンは、ドメインのトラフィックを、例えば Amazon EC2 インスタンスや CloudFront ディストリビューションにルーティングする方法についての情報を保存する場所です。

ドメイン登録、ホストゾーン、ドメインが受信した DNS クエリに関連する料金が発生します。詳細については、「Amazon Route 53 料金表」を参照してください。

> **Note**
>
> ドメインを登録すると、すぐに費用がかかり、元に戻せません。ドメインを自動更新しないことは選択できますが、前払いしてその年の所有権を取得することになります。詳細については、「Amazon Route 53 デベロッパーガイド」の「新しいドメインの登録」を参照してください。

ステップ 1: S3 バケットを作成する。

ストリーミングする元の動画を保存するには、バケットを作成します。

バケットを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [Create bucket (バケットの作成)] を選択します。

   [バケットの作成] ページが開きます。
4. Bucket Name に、バケットの名前 (例: **`tutorial-bucket`**) を入力します。

   Amazon S3 のバケット命名規則の詳細については、「バケットの名前付け」を参照してください。
5. [リージョン] で、バケットを保存する AWS リージョンを選択します。

   可能であれば、ほとんどのビューワーに最も近いリージョンを選択します。バケットのリージョンの詳細については、「バケットの概要」を参照してください。
6. [このバケットのブロックパブリックアクセス設定] が、デフォルト設定 (すべてのパブリックアクセスをブロックが有効) のままであることを確認します。

   すべてのパブリックアクセスをブロック を有効にしても、ビューワーは CloudFront 経由でアップロードされた動画にアクセスできます。この機能は、CloudFront を使用して S3 に保存された動画をホストすることの大きな利点です。

   ユースケースでオフにする必要のある設定が 1 つ以上あることがわかっている場合を除き、すべての設定を有効にしておくことをお勧めします。ブロックパブリックアクセスの詳細については、Amazon S3 ストレージへのパブリックアクセスのブロック を参照してください。

7. 残りの設定はデフォルトのままにしておきます。

   (オプション) 特定のユースケースに追加のバケット設定を設定する場合は、「バケットの作成」を参照してください。

8. [Create bucket (バケットの作成)] を選択します。

ステップ 2: S3 バケットに動画をアップロードする。

次の手順では、コンソールを使用して S3 バケットに動画ファイルをアップロードする方法を示します。S3 に動画ファイルをアップロードするときは、Amazon S3 Transfer Acceleration を使用して、高速かつ安全なファイル転送を設定します。転送アクセラレーションを使用すると、S3 バケットへの動画のアップロードを高速化して、大きな動画の長距離転送を行うことができます。詳細については、「Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定」を参照してください。

バケットにファイルをアップロードするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、ステップ 1 で作成したバケットの名前を選択し (**tutorial-bucket** など)、ファイルをアップロードします。
4. バケットの [オブジェクト] タブで、[アップロード] を選択します。
5. [アップロード] ページの [ファイルとフォルダ] の下で、[ファイルを追加] を選択します。
6. アップロードするファイルを選択し、続いて [オープン] を選択します。

   例えば、`sample.mp4` という名前の動画ファイルをアップロードできます。

7. [アップロード] を選択します。

ステップ 3: CloudFront オリジンアクセスアイデンティティを作成する。

S3 バケットからの動画への直接アクセスを制限するには、オリジンアクセスアイデンティティ (OAI) という特別な CloudFront ユーザーを作成します。このチュートリアルの後半で OAI をディストリビューションに関連付けます。OAI を使用すると、ビューワーは CloudFront をバイパスして、S3 バケットから直接ビデオを取得できません。CloudFront OAI だけが S3 バケット内のファイルにアクセスできます。詳細については、Amazon CloudFront デベロッパーガイド の「オリジンア

クセスアイデンティティを使用して Amazon S3 コンテンツへのアクセスを制限する 」を参照してください。

CloudFront OAI を作成するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 左のナビゲーションペインで、[セキュリティ] セクションの [オリジンアクセス] を選択します。
3. [アイデンティティ] タブで、[オリジンアクセスアイデンティティを作成] を選択します。
4. 新しいオリジンアクセスアイデンティティとして、名前 (例えば、**S3-OAI**) を入力します。
5. [Create] (作成) を選択します。

ステップ 4: CloudFront ディストリビューションを作成する。

CloudFront を使用して S3 バケットで動画を提供および配信するには、CloudFront ディストリビューションを作成する必要があります。

サブステップ

- CloudFront ディストリビューションを作成する
- バケットポリシーの確認

CloudFront ディストリビューションを作成する

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 左のナビゲーションペインで、ディストリビューション を選択します。
3. [ディストリビューションの作成] を選択します。
4. [オリジン] の [オリジンドメイン] で、ステップ 1 で作成した S3 バケットの名前から始まる、S3 オリジンのドメイン名 (例えば、**tutorial-bucket**) を選択します。
5. [オリジンアクセス]では、[レガシーアクセスアイデンティティ] を選択します。
6. [オリジンアクセスアイデンティティ] で、ステップ 3 で作成したオリジンアクセスアイデンティティを選択します (例えば、**S3-OAI**)。
7. [バケットポリシー] で、[はい、バケットポリシーを更新します] を選択します。

8. [ビューワープロトコルポリシー] の [デフォルトのキャッシュ動作] セクションで、[HTTP から HTTPS へリダイレクト] を選択します。

   この機能を選択する際は、HTTP リクエストは HTTPS に自動的にリダイレクトされ、ウェブサイトを保護し、ビューワーのデータを保護します。

9. [Default Cache Behavior(キャッシュ動作のデフォルト)] セクションの他の設定については、デフォルト値を使用します。

   (オプション) CloudFront が別のリクエストをオリジンに転送するまでにファイルを CloudFront キャッシュに保持する期間をコントロールできます。この期間を短くすると、動的なコンテンツを供給できます。この期間を長くすると、ユーザー側のパフォーマンスは向上します。ファイルがエッジキャッシュから直接返される可能性が高くなるためです。期間を長くすると、オリジンの負荷も軽減されます。詳細については、Amazon CloudFront デベロッパーガイドの「コンテンツがエッジキャッシュに保持される期間の管理 (有効期限)」を参照してください。

10. その他のセクションでは、残りの設定はデフォルトのままにしておきます。

    異なる設定オプションの詳細については、Amazon CloudFront デベロッパーガイドの「ディストリビューションを作成または更新する場合に指定する値」を参照してください。

11. ページの最下部で、[ディストリビューションの作成] を選択します。

12. CloudFront ディストリビューションの [全般] タブの [詳細] で、ディストリビューションの [最終変更] 列の値が、[デプロイ] からディストリビューションが最後に変更されたタイムスタンプに変更されます。これには通常数分かかります。

バケットポリシーの確認

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、CloudFront ディストリビューションのオリジンとして上記で使用したバケットの名前を選択します (例えば、**`tutorial-bucket`**)。
4. [アクセス許可] タブを選択します。
5. [バケットポリシー] セクションのバケットポリシーテキストで、次のようなステートメントが表示されることを確認します。

```
{
    "Version": "2008-10-17",
```

```
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity EH1HDMB1FH2TC"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::tutorial-bucket/*"
        }
    ]
}
```

これは、上記の [はい、バケットポリシーを更新します] を選択した際に、CloudFront ディストリビューションがバケットポリシーに追加したステートメントです。

このバケットポリシーの更新は、S3 バケットへのアクセスを制限するように CloudFront ディストリビューションを正常に構成したことを示します。この制限のため、バケット内のオブジェクトには CloudFront ディストリビューションからのみアクセスできます。

ステップ 5: CloudFront ディストリビューション経由で動画にアクセスする。

これで、CloudFront は S3 バケットに保存された動画を提供できます。CloudFront 経由で動画にアクセスするには、CloudFront ディストリビューションのドメイン名を S3 バケット内の動画へのパスの中に含めます。

CloudFront ディストリビューションドメイン名を使用して S3 動画の URL を作成するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 左のナビゲーションペインで、[ディストリビューション] を選択します。
3. ディストリビューションのドメイン名を取得するには、次の手順を実行します。

    a. [オリジン] 列で、ステップ 1 で作成した S3 バケットで始まるオリジン名を見つけて、CloudFront ディストリビューションを識別します (例えば、**tutorial-bucket**)。

   b. リストからディストリビューションを見つけたら、[ドメイン名] 列を広げて、CloudFront ディストリビューションのドメイン名の値をコピーします。

4. 新しいブラウザタブで、上記でコピーしたディストリビューションドメイン名を貼り付けます。
5. 前のブラウザタブに戻り、https://console.aws.amazon.com/s3/ にある S3 コンソールを開きます。
6. 左側のナビゲーションペインで、[バケット] を選択します。
7. [バケット] リストで、ステップ 1 で作成したバケットの名前を選択します (例えば、**tutorial-bucket**)。
8. [オブジェクト] リストで、ステップ 2 でアップロードした動画の名前を選択します (例えば、`sample.mp4`)。
9. オブジェクト詳細ページで、オブジェクトの概要 セクションで、[キー] の値をコピーします。この値は、S3 バケット内でアップロードされたビデオオブジェクトへのパスです。
10. 以前にディストリビューションのドメイン名を貼り付けたブラウザタブに戻り、スラッシュ (**/**) を配信ドメイン名の後に入力し、前にコピーした動画へのパスを貼り付けます (例: `sample.mp4`)。

    これで、S3 動画はパブリックにアクセスでき、CloudFront 経由でホストされた URL は、次のようになります。

    ```
    https://CloudFront distribution domain name/Path to the S3 video
    ```

    `CloudFront ################` および *`S3 ######`* を適切な値で置き換えます。URL の例は **`https://d111111abcdef8.cloudfront.net/sample.mp4`** です。

ステップ 6: カスタムドメイン名を使用するように CloudFront ディストリビューションを設定する。

URL で CloudFront ドメイン名の代わりに独自のドメイン名を使用して S3 動画にアクセスするには、代替ドメイン名を CloudFront ディストリビューションに追加する必要があります。

サブステップ

- SSL 証明書をリクエストする。
- CloudFront ディストリビューションに代替ドメイン名を追加します。

- 代替ドメイン名のトラフィックを CloudFront ディストリビューションのドメイン名にルーティングする DNS レコードを作成する。
- ディストリビューションに対して IPv6 が有効になっているかどうかを確認し、必要に応じて別の DNS レコードを作成します。

SSL 証明書をリクエストする。

ビューワーが動画ストリーミングの URL で HTTPS とカスタムドメイン名を使用できるようにするには、AWS Certificate Manager (ACM) を使用して、Secure Sockets Layer (SSL) 証明書を要求します。SSL 証明書は、ウェブサイトへの暗号化されたネットワーク接続を確立します。

1. AWS Management Console にサインインして、ACM コンソール (https://console.aws.amazon.com/acm/home) を開きます。
2. 入門者向けページが表示されたら、[証明書のプロビジョニング] で [開始] を選択します。
3. [証明書のリクエスト] ページで、[パブリック証明書のリクエスト] を選択し、[証明書のリクエスト] を選択して続行します。
4. [ドメイン名の追加] ページで、SSL/TLS 証明書を使用して保護するサイトの完全修飾ドメイン名を入力します。アスタリスク (*) を使用して、同じドメイン内の複数のサイトを保護するワイルドカード証明書をリクエストします。このチュートリアルでは、[前提条件] で設定したカスタムドメイン名を入力します。例えば、**`*.example.com`** を使用し、[Next (次へ)] を選択します。

   詳細については、AWS Certificate Managerユーザーガイドの「ACM パブリック証明書をリクエストする (コンソール)」を参照してください。
5. [検証方法の選択] ページで、[DNS での検証] を選択します。[次へ] を選択します。

   DNS 設定を編集できる場合は、E メール検証ではなく DNS ドメイン検証を使用することをお勧めします。DNS 検証には E メール検証と比べていくつかの利点があります。詳細については、AWS Certificate Managerユーザーガイドのオプション 1: DNS 検証を参照してください。
6. (オプション) [タグの追加] ページで、メタデータを用いて証明書にタグを付けることができます。
7. [Review] (レビュー) を選択します。
8. [確認] ページで、ドメイン名および検証方法にある情報が正しいことを確認します。次に、[確認してリクエストする] を選択します。

   [検証] ページには、リクエストが処理中で、証明書ドメインが検証中であることが示されます。検証を待機している証明書は、検証保留中状態にあります。

9. [検証] ページで、カスタムドメイン名の左側にある下向き矢印を選択し、[Route 53 でレコードを作成する] を選択して、DNS によるドメインの所有権を検証します。

   これは、AWS Certificate Manager によって DNS 設定に提供される CNAME レコードを追加します。

10. [Route 53 でレコードを作成する] ダイアログボックスで、[作成] を選択します。

    これで、[検証] ページの一番下に、[成功] のステータス通知が表示されるようになりました。

11. [続行] を選択して、[証明書] リストページを表示します。

    新しい証明書のステータスは、30 分以内に [検証保留中] から [発行済み] に変わります。

CloudFront ディストリビューションに代替ドメイン名を追加します。

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 左のナビゲーションペインで、[ディストリビューション] を選択します。
3. ステップ 4 で作成したディストリビューションの ID を選択します。
4. [全般] タブで、設定セクションに移動し、編集を選択します。
5. 編集設定ページの [代替ドメイン名 (CNAME) - オプション] で、[項目を追加] を選択して、この CloudFront ディストリビューションによって提供される S3 動画の URL で使用するカスタムドメイン名を追加します。

   このチュートリアルでは、例えば、`www.example.com` などのサブドメインのトラフィックをルーティングする場合、サブドメイン名 (`www`) とドメイン名 (`example.com`) を入力します。具体的には、**`www.example.com`** を入力します。

   > **Note**
   >
   > 追加する代替ドメイン名 (CNAME) は、CloudFront ディストリビューションに以前アタッチした SSL 証明書の対象である必要があります。

6. カスタム SSL 証明書 - オプション で、上記でリクエストした SSL 証明書を選択します (例えば、**`*.example.com`**)。

> **Note**
>
> SSL 証明書をリクエストした直後に SSL 証明書が表示されない場合は、SSL 証明書を選択できるようになるまで 30 分間待機してから一覧を更新できます。

7. 残りの設定はデフォルト値のままにしておきます。[Save changes] (変更の保存) をクリックします。
8. ディストリビューションの [全般] タブで、ディストリビューションの [最終変更] の値が [デプロイ] からディストリビューションが最後に変更されたタイムスタンプに変更されるまで待機します。

代替ドメイン名のトラフィックを CloudFront ディストリビューションのドメイン名にルーティングする DNS レコードを作成する。

1. AWS Management Console にサインインし、Route 53 コンソール (https://console.aws.amazon.com/route53/) を開きます。
2. 左のナビゲーションペインで [ホストゾーン] を選択します。
3. ホストゾーン ページの 前提条件 で Route 53 が作成したホストゾーンの名前を選択します (例えば、**example.com**)。
4. レコードを作成するを選択し、レコードのクイック作成方法を使用します。
5. [レコード名] で、レコード名の値を、上記で追加した CloudFront ディストリビューションの代替ドメイン名と同じにします。

   このチュートリアルでは、www.example.com などのサブドメインにトラフィックをルーティングするには、ドメイン名なしでサブドメイン名を入力します。例えば、カスタムドメイン名の前にあるテキストフィールドに「**www**」を入力します。
6. [レコードタイプ] で、[A - IPv4 アドレスと一部の AWS リソースにトラフィックをルーティング] を選択します。
7. [値] で、エイリアスリソースを有効にするには、エイリアス切り替えを選択します。
8. [トラフィックのルーティング先] で、[CloudFront ディストリビューションへのエイリアス] を選択します。
9. [ディストリビューションの選択] という検索ボックスで、ステップ 4 で作成した CloudFront ディストリビューションのドメイン名を選択します。

CloudFront ディストリビューションのドメイン名を検索するには、以下の手順を実行します。

a. 新しいブラウザタブで、AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v3/home で CloudFront コンソールを開きます。

b. 左のナビゲーションペインで、[ディストリビューション] を選択します。

c. [オリジン] 列で、ステップ 1 で作成した S3 バケットで始まるオリジン名を見つけて、CloudFront ディストリビューションを識別します (例えば、**tutorial-bucket**)。

d. リストからディストリビューションを見つけたら、ドメイン名 列を広げて、CloudFront ディストリビューションのドメイン名の値を確認します。

10. Route 53 コンソールの [レコードを作成する] ページで、残りの設定はデフォルトのままにしておきます。

11. [レコードを作成] を選択します。

ディストリビューションに対して IPv6 が有効になっているかどうかを確認し、必要に応じて別の DNS レコードを作成します。

ディストリビューションに対して IPv6 が有効になっている場合は、別の DNS レコードを作成する必要があります。

1. ディストリビューションに対して IPv6 が有効になっているかどうかを確認するには、次の手順を実行します。

   a. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。

   b. 左のナビゲーションペインで、[ディストリビューション] を選択します。

   c. ステップ 4 で作成した CloudFront ディストリビューションの ID を選択します。

   d. [全般] タブの [設定] で、IPv6 が有効になっているか確認します。

      ディストリビューションに対して IPv6 が有効になっている場合は、別の DNS レコードを作成する必要があります。

2. ディストリビューションに対して IPv6 が有効になっている場合は、以下を実行して DNS レコードを作成する必要があります。

   a. AWS Management Console にサインインし、Route 53 コンソール (https://console.aws.amazon.com/route53/) を開きます。

b. 左のナビゲーションペインで [ホストゾーン] を選択します。
c. ホストゾーン ページの 前提条件 で Route 53 が作成したホストゾーンの名前を選択します (例えば、**example.com**)。
d. レコードを作成するを選択し、レコードのクイック作成方法を使用します。
e. [レコード名] で、カスタムドメイン名の前のテキストフィールドに、上記の IPv4 DNS レコードを作成したときに入力したのと同じ値を入力します。例えば、このチュートリアルでは、サブドメイン `www.example.com` のトラフィックをルーティングするには、**www** のみを入力します。
f. [レコードタイプ] で、[AAAA - IPv6 アドレスと一部の AWS リソースにトラフィックをルーティング] を選択します。
g. [値] で、エイリアスリソースを有効にするには、エイリアス切り替えを選択します。
h. [トラフィックのルーティング先] で、[CloudFront ディストリビューションへのエイリアス] を選択します。
i. [ディストリビューションの選択] という検索ボックスで、ステップ 4 で作成した CloudFront ディストリビューションのドメイン名を選択します。
j. 残りの設定はデフォルトのままにしておきます。
k. [レコードを作成] を選択します。

ステップ 7: カスタムドメイン名を使用して CloudFront ディストリビューションを介して S3 動画にアクセスする。

カスタム URL を使用して S3 動画にアクセスするには、代替ドメイン名と S3 バケット内の動画へのパスを組み合わせる必要があります。

CloudFront ディストリビューションを介して S3 動画にアクセスするためのカスタム URL を作成するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 左のナビゲーションペインで、[ディストリビューション] を選択します。
3. CloudFront ディストリビューションの代替ドメイン名を取得するには、以下を実行します。
   a. [オリジン] 列で、ステップ 1 で作成した S3 バケットのバケット名で始まるオリジン名を見つけて、CloudFront ディストリビューションを識別します (例えば、**tutorial-bucket**)。

b. リストからディストリビューションを見つけたら、代替ドメイン名 列を広げて、CloudFront ディストリビューションの代替ドメイン名の値をコピーします。

4. 新しいブラウザタブに、CloudFront ディストリビューションの代替ドメイン名を貼り付けます。
5. 前のブラウザタブに戻り、https://console.aws.amazon.com/s3/ にある Amazon S3 コンソールを開きます。
6. ステップ 5 で説明した S3 動画へのパスを見つけます。
7. 以前に代替ドメイン名を貼り付けたブラウザタブに戻り、スラッシュ (**/**) を入力し、S3 動画へのパスを貼り付けます (例えば、`sample.mp4`)。

   これで、S3 動画はパブリックにアクセス可能になり、CloudFront を介して次のようなカスタム URL でホストされます。

   ```
   https://CloudFront distribution alternate domain name/Path to the S3 video
   ```

   *CloudFront ####################*および *S3 #######*適切な値で置き換えます。URL の例は **https://www.example.com/sample.mp4** です。

(オプション)ステップ 8: CloudFront ディストリビューションが受信したリクエストに関するデータを表示する

CloudFront ディストリビューションが受信したリクエストに関するデータを表示するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. CloudFront コンソールの左側のナビゲーションペインで、[レポートと分析] で、コンソールから、キャッシュ統計、人気のあるオブジェクト、上位リファラー、使用状況、ビューワーなどのレポートを選択します。

   各レポートダッシュボードをフィルタリングできます。詳細については、Amazon CloudFront デベロッパーガイドの「コンソール内の CloudFront レポート」セクションを参照してください。
3. データをフィルタリングするには、ステップ 4 で作成した CloudFront ディストリビューションの ID を選択します。

ステップ 9: クリーンアップする。

学習演習としてのみ、CloudFront と Route 53 を使用して S3 ストリーミング動画をホストした場合は、割り当てた AWS リソースを削除して、料金が発生しないようにします。

> **Note**
>
> ドメインを登録すると、すぐに費用がかかり、元に戻せません。ドメインを自動更新しないことは選択できますが、前払いしてその年の所有権を取得することになります。詳細については、「Amazon Route 53 デベロッパーガイド」の「新しいドメインの登録」を参照してください。

サブステップ

- CloudFront ディストリビューションの削除
- DNS レコードの削除
- カスタムドメインのパブリックホストゾーンの削除
- Route 53 からカスタムドメイン名を削除します。
- S3 ソースバケットの元の動画を削除する。
- S3 ソースバケットを削除する

CloudFront ディストリビューションの削除

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 左のナビゲーションペインで、[ディストリビューション] を選択します。
3. [オリジン] 列で、ステップ 1 で作成した S3 バケットのバケット名で始まるオリジン名を見つけて、CloudFront ディストリビューションを識別します (例えば、**`tutorial-bucket`**)。
4. CloudFront ディストリビューションを削除するには、まずディストリビューションを無効にする必要があります。
   - [ステータス] 列の値が [有効] で、[最終変更] の値がディストリビューションが最後に変更されたときのタイムスタンプである場合は、ディストリビューションを無効にしてから削除してください。

- [ステータス] の値が [有効] で、[最終変更] の値がデプロイの場合、[ステータス] がディストリビューションが最後に変更されたタイムスタンプに変わるまで待ちます。次に、ディストリビューションを無効にしてから、ディストリビューションを削除します。

5. CloudFront ディストリビューションを無効にするには、次の手順を実行します。

   a. ディストリビューションリストで、削除するディストリビューションの ID の横にあるチェックボックスをオンにします。

   b. ディストリビューションを無効にするには、[無効] を選択し、[無効] を選択して確定します。

      代替ドメイン名が関連付けられているディストリビューションを無効にすると、CloudFront は、別のディストリビューションに同じドメイン (`*.example.com` など) と一致するワイルドカード(*) 付きの代替ドメイン名がある場合でも、このドメイン名 (`www.example.com` など) へのトラフィックの受信を停止します。

   c. [状態] 列の値が直ちに [無効] に変わります。[最終変更] の値が [デプロイ] からディストリビューションが最後に変更されたタイムスタンプに変更されるまで待機します。

      CloudFront はこの変更をすべてのエッジロケーションに伝達する必要があるため、この更新が完了してディストリビューションを削除できるようになるまでに数分かかることがあります。

6. 無効になっているディストリビューションを削除するには、次の手順を実行します。

   a. 削除するディストリビューションの ID の横にあるチェックボックスをオンにします。

   b. 削除 を選択し、削除 を選択して確認します。

DNS レコードの削除

ドメインのパブリックホストゾーン (DNS レコードを含む) を削除する場合は、Amazon Route 53 デベロッパーガイドの カスタムドメインのパブリックホストゾーンの削除 を参照してください。 ステップ 6 で作成された DNS レコードを削除するだけの場合は、次の操作を行います。

1. AWS Management Console にサインインし、Route 53 コンソール (https://console.aws.amazon.com/route53/) を開きます。
2. 左のナビゲーションペインで [ホストゾーン] を選択します。
3. ホストゾーン ページの 前提条件 で Route 53 が作成したホストゾーンの名前を 選択します (例えば、**`example.com`**)。

4. レコードのリストで、削除するレコード (ステップ 6 で作成したレコード) の横にあるチェックボックスをオンにします。

> **Note**
>
> [タイプ] の値が [NS] または [SOA] のレコードを削除することはできません。

5. [レコードセットの削除] を選択します。
6. 削除を確認するには、[削除] を選択します。

レコードの変更が Route 53 DNS サーバーに伝達されるまでにしばらく時間がかかります。現在、変更が反映されたことを確認するには、GetChange API アクションを使用する方法しかありません。通常、変更は 60 秒以内にすべての Route 53 ネームサーバーに伝播されます。

カスタムドメインのパブリックホストゾーンの削除

> **Warning**
>
> ドメイン登録は維持するものの、ウェブサイトやウェブアプリケーションへのインターネットトラフィックのルーティングを停止する場合、ホストゾーンを削除する代わりに、ホストゾーン (上記) 内のレコードを削除することをお勧めします。
> さらに、ホストゾーンを削除すると、他のユーザーがお客様のドメイン名を使用してドメインを使って自分のリソースにトラフィックをルーティングする可能性があります。
> さらに、ホストゾーンを削除した場合、復元することはできません。新しいホストゾーンを作成して、ドメイン登録のネームサーバーを更新する必要があります。更新が有効になるには、最大 48 時間かかることがあります。
> ドメインをインターネット上で利用できなくするには、DNS サービスを無料の DNS サービスに移行し、Route 53 のホストゾーンを削除することをお勧めします。これにより、今後 DNS クエリが誤ってルーティングされることを防ぐことができます。
>
> 1. ドメインが Route 53 に登録されている場合に、Route 53 ネームサーバーを新しい DNS サービスのネームサーバーで置き換える情報については、Amazon Route 53 デベロッパーガイドで、ドメインのネームサーバーとグルーレコードの追加または変更を参照してください。
> 2. ドメインが他のレジストラに登録されている場合、レジストラが提供する方法を使用してドメインのネームサーバーを変更します。

> **Note**
>
> サブドメイン (`www.example.com`) のホストゾーンを削除する場合は、ドメイン (`example.com`) のネームサーバーを変更する必要はありません。

1. AWS Management Console にサインインし、Route 53 コンソール (https://console.aws.amazon.com/route53/) を開きます。
2. 左のナビゲーションペインで [ホストゾーン] を選択します。
3. [ホストゾーン] ページで、削除するホストゾーンの名前を選択します。
4. ホストゾーンの [レコード] タブで、削除するホストゾーンに NS レコードと SOA レコードのみが含まれていることを確認します。

   他のレコードが含まれている場合は、それらを削除します。

   ホストゾーンでサブドメインの NS レコードを作成した場合は、それらのレコードも削除します。
5. ホストゾーンの DNSSEC 署名 タブで、DNSSEC 署名が有効になっている場合は無効にします。詳細については、「 Amazon Route 53 デベロッパーガイド」の「DNSSEC 署名の無効」を参照してください。
6. ホストゾーンの詳細ページの上部で、[ゾーンを削除] を選択します。
7. **`delete`** と入力して削除を確認し、[Delete (削除)] を選択します。

Route 53 からカスタムドメイン名を削除します。

最上位ドメイン (TLD) では、必要がなくなった登録を削除できます。登録が期限切れになる前に Route 53 からドメイン名登録を削除した場合でも、登録料は AWS から払い戻しされません。詳細については、Amazon Route 53 デベロッパーガイドの「ドメイン名の登録を削除する」を参照してください。

> **Important**
>
> AWS アカウント 間でドメインを移管する場合や、他のレジストラにドメインを移管する場合は、ドメインを削除せずにすぐに登録が予測されます。代わりに、「Amazon Route 53 デベロッパーガイド」で該当するドキュメントを参照してください。
>
> - 異なる AWS アカウント へのドメインの移管
> - Amazon Route 53 から別のレジストラへのドメインの移行

S3 ソースバケットの元の動画を削除する。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット名] リストで、ステップ 2 で動画をアップロードしたバケットの名前を選択します (例えば、**tutorial-bucket**)。
4. オブジェクト タブで、削除するオブジェクトの名前の左にあるチェックボックスをオンにします (例えば、`sample.mp4`)。
5. [削除] を選択します。
6. [オブジェクトを完全に削除しますか?] で、**permanently delete** と入力し、このオブジェクトを削除することを確定します。
7. [オブジェクトの削除] を選択します。

S3 ソースバケットを削除する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、ステップ 1 で作成したバケットの名前の横にあるオプションボタンを選択します (例えば、**tutorial-bucket**)。
4. [削除] を選択します。
5. [バケットの削除] ページで、テキストフィールドにバケット名を入力することでバケットを削除することを確認し、[バケットを削除] を選択します。

次のステップ

このチュートリアルを完了すると、次の関連するユースケースをさらに調べることができます。

- CloudFront ディストリビューションでこれらの動画をホストする前に、特定のテレビや接続されたデバイスで必要なストリーミング形式に S3 動画をトランスコードします。

  Amazon S3 バッチオペレーション、AWS Lambda、および AWS Elemental MediaConvert を使用して、動画のコレクションをさまざまな出力メディア形式にバッチトランスコードする方法の詳細については、「チュートリアル: S3 バッチオペレーション、AWS Lambda、および AWS Elemental MediaConvert を使用した動画のバッチトランスコーディング」を参照してください。
- CloudFront と Route 53 を使用して、画像、オーディオ、モーショングラフィックス、スタイルシート、HTML、JavaScript、React アプリなど、S3 に保存されている他のオブジェクトをホストします。

  例については、「チュートリアル: Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定」および「Amazon CloudFront によるウェブサイトの高速化」を参照してください。
- Amazon S3 Transfer Acceleration を使用して、高速かつ安全なファイル転送を設定します。転送アクセラレーションを使用すると、S3 バケットへの動画のアップロードを高速化して、大きな動画の長距離転送を行うことができます。Transfer Acceleration は、CloudFront のグローバルに分散したエッジロケーションや AWS バックボーンネットワークを経由してトラフィックをルーティングすることで、転送パフォーマンスを向上させます。また、ネットワークプロトコルの最適化も利用します。詳細については、「Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定」を参照してください。

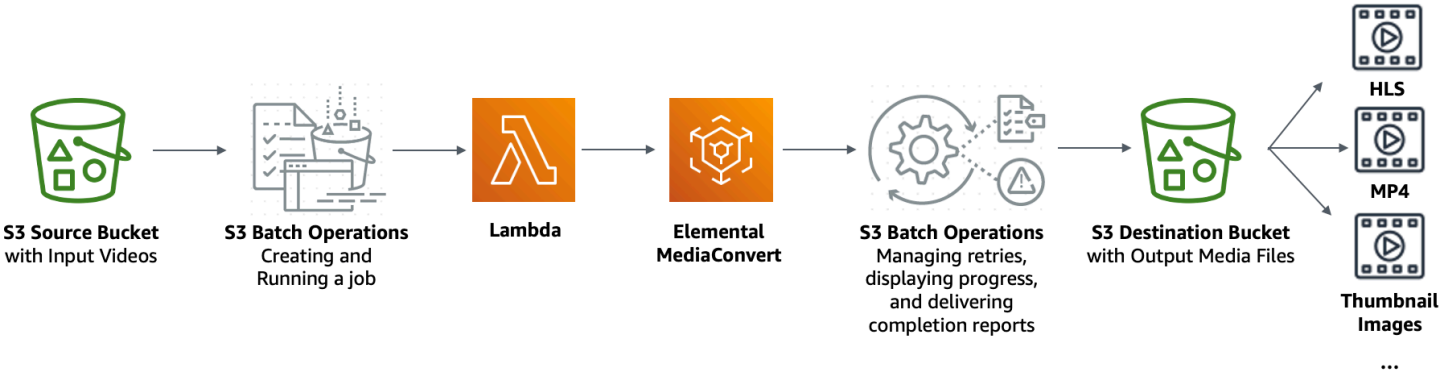
チュートリアル: S3 バッチオペレーション、AWS Lambda、および AWS Elemental MediaConvert を使用した動画のバッチトランスコーディング

動画消費者は、あらゆる形状、サイズ、ビンテージのデバイスを使用してメディアコンテンツを楽しんでいます。この幅広い一連のデバイスは、コンテンツ作成者やディストリビューターにとって課題となります。動画は、ワンサイズですべてにフィットするフォーマットではなく、幅広いサイズ、フォーマット、ビットレートに変換する必要があります。変換が必要な動画が多数ある場合、この変換タスクはさらに困難になります。

AWS では、以下を実行するスケーラブルな分散アーキテクチャを構築する方法を提供します。

- 入力動画を取り込みます。
- さまざまなデバイスで再生するために動画を処理します。
- トランスコードされたメディアファイルを格納します。
- 需要に合わせて出力メディアファイルを配信します。

Amazon S3 に広範な動画リポジトリを保存している場合は、これらの動画をソース形式から、特定の動画プレーヤーやデバイスに必要なサイズ、解像度、形式の複数のファイルタイプに変換できます。具体的には、S3 バッチ操作 は、S3 ソースバケット内の既存の入力動画の AWS Lambda 関数を呼び出すためのソリューションを提供します。次に、Lambda 関数は AWS Elemental MediaConvert を呼び出して、広範な動画トランスコーディングタスクを実行します。変換後の出力メディアファイルは S3 保存先バケットに保存されます。



目的

このチュートリアルでは、S3 送信先バケットに保存された動画のバッチトランスコーディング用に Lambda 関数を呼び出すように S3 バッチ操作を設定する方法を学習します。Lambda 関数は MediaConvert を呼び出して、動画をトランスコードします。S3 ソースバケット内の各動画の出力は、次のとおりです。

- 複数のサイズのデバイスおよびさまざまな帯域幅で再生するための HTTP Live Streaming (HLS) アダプティブビットレートストリーム。
- MP4 動画ファイル
- サムネイル画像は、間隔をおいて収集されます。

トピック

- 前提条件

前提条件

このチュートリアルを開始する前に、トランスコードされる動画がすでにその中に保存されている Amazon S3 ソースバケットが必要です (例、**`tutorial-bucket-1`**)。

必要に応じて、バケットに別の名前を付けることができます。Amazon S3 のバケット命名規則の詳細については、「バケットの名前付け」を参照してください。

S3 ソースバケットの場合、このバケットのブロックパブリックアクセス設定 に関連する設定をデフォルトに設定したままにします (すべて のパブリックアクセスをブロックする が有効)。詳細については、「バケットの作成」を参照してください。

S3 ソースバケットへの動画のアップロードの詳細については、オブジェクトのアップロード を参照してください。S3 に動画ファイルをアップロードするときは、Amazon S3 Transfer Acceleration を使用して、高速かつ安全なファイル転送を設定します。転送アクセラレーションを使用すると、S3 バケットへの動画のアップロードを高速化して、大きな動画の長距離転送を行うことができます。詳細については、「Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定」を参照してください。

ステップ 1: 出力メディアファイルの S3 バケットを作成する。

このステップでは、変換後の出力メディアファイルを保存するための S3 ソースバケットを作成します。また、Cross Origin Resource Sharing (CORS) 設定を作成して、S3 の送信先バケットに保存されているトランスコードされたメディアファイルへのクロスオリジンアクセスを許可します。

サブステップ

- 出力メディアファイル用のバケットを作成する。
- CORS 設定を S3 出力バケットに追加する

出力メディアファイル用のバケットを作成する。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [Create bucket (バケットの作成)] を選択します。
4. Bucket Name に、バケットの名前 (例: **`tutorial-bucket-2`**) を入力します。
5. [リージョン] で、バケットを保存する AWS リージョンを選択します。
6. 出力メディアファイルへのパブリックアクセスを確保するには、Block Public Access settings (パブリックアクセスのブロック設定) で、Block all public access (すべてのパブリックアクセスをブロック) をオフにします。

   > **Warning**
   >
   > このステップを完了する前に「Amazon S3 ストレージへのパブリックアクセスのブロック」を読んで、パブリックアクセスを許可することに伴うリスクを理解し、了承します。ブロックパブリックアクセス設定をオフにしてバケットをパブリックにすると、インターネット上の誰でもバケットにアクセスできるようになります。バケットへのすべてのパブリックアクセスをブロックすることをお勧めします。
   > ブロックパブリックアクセス設定をクリアしたくない場合は、Amazon CloudFront を使用して、トランスコードされたメディアファイルをビューワー (エンドユーザー) に配信できます。詳細については、「チュートリアル: Amazon S3、Amazon CloudFront、Amazon Route 53 を使用したオンデマンドストリーミング動画のホスティング。」を参照してください。

7. [現在の設定により、このバケットと保存されたオブジェクトがパブリックになる可能性があることを了承します] の横にあるチェックボックスを選択します。
8. 残りの設定はデフォルト値のままにしておきます。
9. [Create bucket (バケットの作成)] を選択します。

CORS 設定を S3 出力バケットに追加する

JSON CORS 設定は、特定のドメインにロードされたクライアントウェブアプリケーション (このコンテキストでは動画プレーヤー) が、異なるドメインでトランスコードされた出力メディアファイルを再生する方法を定義します。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. バケット リストで、先に作成したバケットの名前を選択します(例: **tutorial-bucket-2**) 。
4. [アクセス許可] タブを選択します。
5. [CORS (クロスオリジンリソース共有)] セクションで、[編集] を選択します。
6. [CORS 設定] テキストボックスで、次に示す CORS 設定をコピーして貼り付けます。

   CORS 設定は、JSON 形式である必要があります。この例では、AllowedOrigins 属性はワイルドカード文字 (*) を使用して、すべてのオリジンを指定します。特定のオリジンがわかっている場合は、AllowedOrigins 属性を特定のプレーヤーの URL に限定します。この属性およびその他の属性の設定の詳細については、「CORS の設定」を参照してください。

   ```
   [
       {
           "AllowedOrigins": [
               "*"
           ],
           "AllowedMethods": [
               "GET"
           ],
           "AllowedHeaders": [
               "*"
           ],
           "ExposeHeaders": []

       }
   ]
   ```

7. [Save changes] (変更の保存) をクリックします。

ステップ 2: MediaConvert 用に IAM ロールを作成する

AWS Elemental MediaConvert を使用して S3 バケットに保存されている入力動画をトランスコーディングするには、AWS Identity and Access Management (IAM) サービスロールを使用して、S3 ソースバケットおよび宛先バケットから動画ファイルを読み書きするための MediaConvert アクセス許可を付与します。トランスコーディングジョブを実行すると、MediaConvert コンソールはこのロールを使用します。

MediaConvert の IAM ロールを作成するには

1. 選択したロール名を使用して IAM ロールを作成します (例: **`tutorial-mediaconvert-role`**)。このロールを作成するには、「AWS Elemental MediaConvert ユーザーガイド」の「IAM (コンソール) で MediaConvert ロールを作成する」に記載のステップに従ってください。
2. MediaConvert の IAM ロールを作成した後、ロール のリストで、作成した MediaConvert のロールの名前を選択します (例: **`tutorial-mediaconvert-role`**)。
3. 概要 ページで、`arn:aws:iam::` で始まる ロール ARN をコピーし、後で使用できるように ARN を保存します。

   ARN の詳細については、AWS 全般のリファレンスの Amazon リソースネーム (ARN) を参照してください。

ステップ 3: Lambda 関数の IAM ロールを作成する

MediaConvert および S3 バッチ操作で動画をバッチトランスコードするには、これらの 2 つのサービスを接続して動画を変換する Lambda 関数を使用します。この Lambda 関数には、MediaConvert および S3 バッチ操作にアクセスするための許可を Lambda 関数に付与する IAM ロールが必要です。

サブステップ

- Lambda 関数の IAM ロールを作成する
- Lambda 関数の IAM ロールにインラインポリシーを埋め込む

Lambda 関数の IAM ロールを作成する

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

2. 左側のナビゲーションペインから、[Roles (ロール)] を選択し、[Create role (ロールの作成)] を選択します。
3. AWS サービス ロールの種類を選択し、一般的ユースケース で、Lambda を選択します。
4. [Next: Permissions] (次へ: アクセス許可) を選択します。
5. [Attach Permissions policies (許可ポリシーのアタッチ)]ページにあるポリシーのフィルタリング に **`AWSLambdaBasicExecutionRole`** を入力します。管理ポリシー AWSLambdaBasicExecutionRole をこのロールにアタッチして、Amazon CloudWatch Logs への書き込み許可を付与するには、AWSLambdaBasicExecutionRole の横にあるチェックボックスを選択します。
6. [Next: Tags] (次へ: タグ) を選択します。
7. (オプション) 管理ポリシーにタグを追加します。
8. [次へ: レビュー] を選択します。
9. [ロール名] に「**`tutorial-lambda-transcode-role`**」と入力します。
10. [ロールの作成] を選択します。

Lambda 関数の IAM ロールにインラインポリシーを埋め込む

Lambda 関数の実行に必要な MediaConvert リソースに許可を付与するためには、インラインポリシーを使用する必要があります。

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。
3. ロール のリストで、Lambda 関数用に先に作成した IAM ロールの名前を選択します (例、**`tutorial-lambda-transcode-role`**)。
4. [アクセス許可] タブを選択します。
5. [Add inline policy] を選択します。
6. [JSON] タブを選択し、以下の JSON ポリシーをコピーペーストします。

   JSON ポリシーでは、`Resource` の例の ARN 値を、ステップ 2 で作成した MediaConvert の IAM ロールのロール ARN に置き換えます (例、**`tutorial-mediaconvert-role`**)。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "Logging"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/tutorial-mediaconvert-role"
            ],
            "Effect": "Allow",
            "Sid": "PassRole"
        },
        {
            "Action": [
                "mediaconvert:*"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "MediaConvertService"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "S3Service"
        }
    ]
}
```

7. [ポリシーの確認] を選択します。

8. [Name] (名前) に **tutorial-lambda-policy** と入力します。
9. [ポリシーの作成] を選択します。

   インラインポリシーを作成すると、Lambda 関数の IAM ロールに自動的に埋め込まれます。

ステップ 4: 動画トランスコーディング用の Lambda 関数の作成

チュートリアルの子のセクションでは、S3 バッチ操作および MediaConvert と統合するには、SDK for Python を使用して Lambda 関数を構築します。S3 ソースバケットにすでに保存されている動画のトランスコーディングを開始するには、S3 ソースバケット内の各動画に対して Lambda 関数を直接呼び出す S3 バッチ操作ジョブを実行します。次に、Lambda 関数は、各動画のトランスコーディングジョブを MediaConvert に送信します。

サブステップ

- Lambda 関数コードを記述し、デプロイパッケージを作成する
- 実行ロール (コンソール) を使用した Lambda 関数の作成
- .zip ファイルアーカイブを使用して Lambda 関数コードをデプロイし、Lambda 関数を設定します (コンソール)。

Lambda 関数コードを記述し、デプロイパッケージを作成する

1. ローカルマシンで、`batch-transcode` という名前のフォルダを作成します。
2. `batch-transcode` フォルダに、JSON ジョブ設定でファイルを作成します。例えば、このセクションで提供されている設定を使用して、ファイル `job.json` に名前を付けることができます。

   `job.json` ファイルでは、以下を指定します。

   - トランスコードするファイル
   - 入力動画のトランスコード方法
   - 作成したい出力メディアファイル
   - トランスコードされたファイルの名前
   - トランスコードされたファイルを保存する場所
   - 適用する高度な機能など

このチュートリアルでは、次の job.json ファイルを使用して、S3 ソースバケットの各動画に対して次の出力を作成します。

- 複数のサイズのデバイスおよびさまざまな帯域幅で再生するための HTTP Live Streaming (HLS) アダプティブビットレートストリーム。
- MP4 動画ファイル
- 間隔を置いて収集されたサムネイル画像

このサンプル job.json ファイルは、品質が定義された可変ビットレート (QVBR) を使用して動画品質を最適化します。HLS 出力は、Apple に準拠しています (動画からのオーディオの混合、セグメント継続時間 (6 秒)、自動 QVBR により最適化された動画品質)。

ここで提供される設定例を使用しない場合は、ユースケースに基づく job.json 仕様を生成します。出力間で一貫性を保つために、入力ファイルの動画とオーディオの設定が似ていることを確認してください。異なる動画およびオーディオ構成を持つ任意の入力ファイルについては、別のオートメーション (一意の job.json 設定) を作成します。詳細については、「AWS Elemental MediaConvert ユーザーガイド」の「JSON での AWS Elemental MediaConvert ジョブ設定の例」を参照してください。

```
{
  "OutputGroups": [
    {
      "CustomName": "HLS",
      "Name": "Apple HLS",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "VideoPid": 481,
```

```
            "AudioPids": [
              482,
              483,
              484,
              485,
              486,
              487,
              488,
              489,
              490,
              491,
              492
            ]
          }
        },
        "VideoDescription": {
          "Width": 640,
          "ScalingBehavior": "DEFAULT",
          "Height": 360,
          "TimecodeInsertion": "DISABLED",
          "AntiAlias": "ENABLED",
          "Sharpness": 50,
          "CodecSettings": {
            "Codec": "H_264",
            "H264Settings": {
              "InterlaceMode": "PROGRESSIVE",
              "NumberReferenceFrames": 3,
              "Syntax": "DEFAULT",
              "Softness": 0,
              "GopClosedCadence": 1,
              "GopSize": 2,
              "Slices": 1,
              "GopBReference": "DISABLED",
              "MaxBitrate": 1200000,
              "SlowPal": "DISABLED",
              "SpatialAdaptiveQuantization": "ENABLED",
              "TemporalAdaptiveQuantization": "ENABLED",
              "FlickerAdaptiveQuantization": "DISABLED",
              "EntropyEncoding": "CABAC",
              "FramerateControl": "INITIALIZE_FROM_SOURCE",
              "RateControlMode": "QVBR",
              "CodecProfile": "MAIN",
              "Telecine": "NONE",
              "MinIInterval": 0,
```

```
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_360"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "TimedMetadataPid": 502,
              "VideoPid": 481,
              "AudioPids": [
                482,
```

```
                483,
                484,
                485,
                486,
                487,
                488,
                489,
                490,
                491,
                492
              ]
            }
          },
          "VideoDescription": {
            "Width": 960,
            "ScalingBehavior": "DEFAULT",
            "Height": 540,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 3500000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
```

```
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_540"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "VideoPid": 481,
              "AudioPids": [
                482,
                483,
                484,
                485,
```

```
                486,
                487,
                488,
                489,
                490,
                491,
                492
              ]
            }
          },
          "VideoDescription": {
            "Width": 1280,
            "ScalingBehavior": "DEFAULT",
            "Height": 720,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 5000000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
```

```
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_720"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {}
          },
          "AudioDescriptions": [
            {
              "AudioSourceName": "Audio Selector 1",
              "CodecSettings": {
                "Codec": "AAC",
                "AacSettings": {
                  "Bitrate": 96000,
                  "CodingMode": "CODING_MODE_2_0",
                  "SampleRate": 48000
                }
              }
            }
          ],
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
```

```
            "AudioTrackType": "ALTERNATE_AUDIO_AUTO_SELECT_DEFAULT"
          }
        },
        "NameModifier": "_audio"
      }
    ],
    "OutputGroupSettings": {
      "Type": "HLS_GROUP_SETTINGS",
      "HlsGroupSettings": {
        "ManifestDurationFormat": "INTEGER",
        "SegmentLength": 6,
        "TimedMetadataId3Period": 10,
        "CaptionLanguageSetting": "OMIT",
        "Destination": "s3://EXAMPLE-BUCKET/HLS/",
        "DestinationSettings": {
          "S3Settings": {
            "AccessControl": {
              "CannedAcl": "PUBLIC_READ"
            }
          }
        },
        "TimedMetadataId3Frame": "PRIV",
        "CodecSpecification": "RFC_4281",
        "OutputSelection": "MANIFESTS_AND_SEGMENTS",
        "ProgramDateTimePeriod": 600,
        "MinSegmentLength": 0,
        "DirectoryStructure": "SINGLE_DIRECTORY",
        "ProgramDateTime": "EXCLUDE",
        "SegmentControl": "SEGMENTED_FILES",
        "ManifestCompression": "NONE",
        "ClientCache": "ENABLED",
        "StreamInfResolution": "INCLUDE"
      }
    }
  },
  {
    "CustomName": "MP4",
    "Name": "File Group",
    "Outputs": [
      {
        "ContainerSettings": {
          "Container": "MP4",
          "Mp4Settings": {
            "CslgAtom": "INCLUDE",
```

```
            "FreeSpaceBox": "EXCLUDE",
            "MoovPlacement": "PROGRESSIVE_DOWNLOAD"
          }
        },
        "VideoDescription": {
          "Width": 1280,
          "ScalingBehavior": "DEFAULT",
          "Height": 720,
          "TimecodeInsertion": "DISABLED",
          "AntiAlias": "ENABLED",
          "Sharpness": 100,
          "CodecSettings": {
            "Codec": "H_264",
            "H264Settings": {
              "InterlaceMode": "PROGRESSIVE",
              "ParNumerator": 1,
              "NumberReferenceFrames": 3,
              "Syntax": "DEFAULT",
              "Softness": 0,
              "GopClosedCadence": 1,
              "HrdBufferInitialFillPercentage": 90,
              "GopSize": 2,
              "Slices": 2,
              "GopBReference": "ENABLED",
              "HrdBufferSize": 10000000,
              "MaxBitrate": 5000000,
              "ParDenominator": 1,
              "EntropyEncoding": "CABAC",
              "RateControlMode": "QVBR",
              "CodecProfile": "HIGH",
              "MinIInterval": 0,
              "AdaptiveQuantization": "AUTO",
              "CodecLevel": "AUTO",
              "FieldEncoding": "PAFF",
              "SceneChangeDetect": "ENABLED",
              "QualityTuningLevel": "SINGLE_PASS_HQ",
              "UnregisteredSeiTimecode": "DISABLED",
              "GopSizeUnits": "SECONDS",
              "ParControl": "SPECIFIED",
              "NumberBFramesBetweenReferenceFrames": 3,
              "RepeatPps": "DISABLED",
              "DynamicSubGop": "ADAPTIVE"
            }
          },
```

```
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "AudioDescriptions": [
            {
              "AudioTypeControl": "FOLLOW_INPUT",
              "AudioSourceName": "Audio Selector 1",
              "CodecSettings": {
                "Codec": "AAC",
                "AacSettings": {
                  "AudioDescriptionBroadcasterMix": "NORMAL",
                  "Bitrate": 160000,
                  "RateControlMode": "CBR",
                  "CodecProfile": "LC",
                  "CodingMode": "CODING_MODE_2_0",
                  "RawFormat": "NONE",
                  "SampleRate": 48000,
                  "Specification": "MPEG4"
                }
              },
              "LanguageCodeControl": "FOLLOW_INPUT",
              "AudioType": 0
            }
          ]
        }
      ],
      "OutputGroupSettings": {
        "Type": "FILE_GROUP_SETTINGS",
        "FileGroupSettings": {
          "Destination": "s3://EXAMPLE-BUCKET/MP4/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          }
        }
      }
    },
    {
      "CustomName": "Thumbnails",
```

```
      "Name": "File Group",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "RAW"
          },
          "VideoDescription": {
            "Width": 1280,
            "ScalingBehavior": "DEFAULT",
            "Height": 720,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "FRAME_CAPTURE",
              "FrameCaptureSettings": {
                "FramerateNumerator": 1,
                "FramerateDenominator": 5,
                "MaxCaptures": 500,
                "Quality": 80
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          }
        }
      ],
      "OutputGroupSettings": {
        "Type": "FILE_GROUP_SETTINGS",
        "FileGroupSettings": {
          "Destination": "s3://EXAMPLE-BUCKET/Thumbnails/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          }
        }
      }
    }
  ],
```

```
  "AdAvailOffset": 0,
  "Inputs": [
    {
      "AudioSelectors": {
        "Audio Selector 1": {
          "Offset": 0,
          "DefaultSelection": "DEFAULT",
          "ProgramSelection": 1
        }
      },
      "VideoSelector": {
        "ColorSpace": "FOLLOW"
      },
      "FilterEnable": "AUTO",
      "PsiControl": "USE_PSI",
      "FilterStrength": 0,
      "DeblockFilter": "DISABLED",
      "DenoiseFilter": "DISABLED",
      "TimecodeSource": "EMBEDDED",
      "FileInput": "s3://EXAMPLE-INPUT-BUCKET/input.mp4"
    }
  ]
}
```

3. batch-transcode フォルダで、Lambda 関数を使用してファイルを作成します。次の Python の例を使用し、ファイル convert.py に名前を付けます。

   S3 バッチ操作は、特定のタスクデータを Lambda 関数に送信 し、結果データを返します。Lambda 関数のリクエストとレスポンスの例、レスポンスコードと結果コード、S3 バッチ操作の Lambda 関数の例に関する詳細は、「AWS Lambda 関数の呼び出し」を参照してください。

```
import json
import os
from urllib.parse import urlparse
import uuid
import boto3

"""
When you run an S3 Batch Operations job, your job
invokes this Lambda function. Specifically, the Lambda function is
invoked on each video object listed in the manifest that you specify
for the S3 Batch Operations job in Step 5.
```

```
Input parameter "event": The S3 Batch Operations event as a request
                         for the Lambda function.

Input parameter "context": Context about the event.

Output: A result structure that Amazon S3 uses to interpret the result
        of the operation. It is a job response returned back to S3 Batch
 Operations.
"""
def handler(event, context):

    invocation_schema_version = event['invocationSchemaVersion']
    invocation_id = event['invocationId']
    task_id = event['tasks'][0]['taskId']

    source_s3_key = event['tasks'][0]['s3Key']
    source_s3_bucket = event['tasks'][0]['s3BucketArn'].split(':::')[-1]
    source_s3 = 's3://' + source_s3_bucket + '/' + source_s3_key

    result_list = []
    result_code = 'Succeeded'
    result_string = 'The input video object was converted successfully.'

    # The type of output group determines which media players can play
    # the files transcoded by MediaConvert.
    # For more information, see Creating outputs with AWS Elemental MediaConvert.
    output_group_type_dict = {
        'HLS_GROUP_SETTINGS': 'HlsGroupSettings',
        'FILE_GROUP_SETTINGS': 'FileGroupSettings',
        'CMAF_GROUP_SETTINGS': 'CmafGroupSettings',
        'DASH_ISO_GROUP_SETTINGS': 'DashIsoGroupSettings',
        'MS_SMOOTH_GROUP_SETTINGS': 'MsSmoothGroupSettings'
    }

    try:
        job_name = 'Default'
        with open('job.json') as file:
            job_settings = json.load(file)

        job_settings['Inputs'][0]['FileInput'] = source_s3

        # The path of each output video is constructed based on the values of
        # the attributes in each object of OutputGroups in the job.json file.
```

```
        destination_s3 = 's3://{0}/{1}/{2}' \
            .format(os.environ['DestinationBucket'],
                    os.path.splitext(os.path.basename(source_s3_key))[0],
                    os.path.splitext(os.path.basename(job_name))[0])

        for output_group in job_settings['OutputGroups']:
            output_group_type = output_group['OutputGroupSettings']['Type']
            if output_group_type in output_group_type_dict.keys():
                output_group_type = output_group_type_dict[output_group_type]
                output_group['OutputGroupSettings'][output_group_type]
['Destination'] = \
                    "{0}{1}".format(destination_s3,
                                    urlparse(output_group['OutputGroupSettings']
[output_group_type]['Destination']).path)
            else:
                raise ValueError("Exception: Unknown Output Group Type {}."
                                 .format(output_group_type))

        job_metadata_dict = {
            'assetID': str(uuid.uuid4()),
            'application': os.environ['Application'],
            'input': source_s3,
            'settings': job_name
        }

        region = os.environ['AWS_DEFAULT_REGION']
        endpoints = boto3.client('mediaconvert', region_name=region) \
            .describe_endpoints()
        client = boto3.client('mediaconvert', region_name=region,
                              endpoint_url=endpoints['Endpoints'][0]['Url'],
                              verify=False)

        try:
            client.create_job(Role=os.environ['MediaConvertRole'],
                              UserMetadata=job_metadata_dict,
                              Settings=job_settings)
        # You can customize error handling based on different error codes that
        # MediaConvert can return.
        # For more information, see MediaConvert error codes.
        # When the result_code is TemporaryFailure, S3 Batch Operations retries
        # the task before the job is completed. If this is the final retry,
        # the error message is included in the final report.
        except Exception as error:
            result_code = 'TemporaryFailure'
```

```
            raise

    except Exception as error:
        if result_code != 'TemporaryFailure':
            result_code = 'PermanentFailure'
        result_string = str(error)

    finally:
        result_list.append({
            'taskId': task_id,
            'resultCode': result_code,
            'resultString': result_string,
        })

    return {
        'invocationSchemaVersion': invocation_schema_version,
        'treatMissingKeyAs': 'PermanentFailure',
        'invocationId': invocation_id,
        'results': result_list
    }
```

4. ローカルターミナルで、convert.py および lambda.zip という名前の .zip ファイルとして job.json を使用してデプロイパッケージを作成するには、上記で作成した batch-transcode フォルダを開き、次のコマンドを実行します。

   macOS ユーザーの場合は、以下のコマンドを使用します。

   ```
   zip -r lambda.zip convert.py job.json
   ```

   Windows ユーザー の場合は、以下のコマンドを実行します。

   ```
   powershell Compress-Archive convert.py lambda.zip
   ```

   ```
   powershell Compress-Archive -update job.json lambda.zip
   ```

実行ロール (コンソール) を使用した Lambda 関数の作成

1. AWS Lambda コンソールを https://console.aws.amazon.com/lambda/ で開きます。
2. 左側のナビゲーションペインで、[関数] を選択します。

3. [Create function (関数の作成)] を選択します。
4. Author from scratch を選択します。
5. 基本的な情報 で、以下の作業を行います。

   a. [関数名] に「**tutorial-lambda-convert**」と入力します。

   b. ランタイム で、Python 3.8 またはそれ以降のバージョンを選択します。
6. デフォルト実行ロールを変更する を選択し、実行ロール で、既存のロールを使用する を選択します。
7. 既存のロール で、ステップ 3 で Lambda 関数用に作成した IAM ロールの名前を選択します (例、**tutorial-lambda-transcode-role**)。
8. 残りの設定はデフォルトのままにしておきます。
9. [Create function (関数の作成)] を選択します。

.zip ファイルアーカイブを使用して Lambda 関数コードをデプロイし、Lambda 関数を設定します (コンソール)。

1. 上で作成した Lambda 関数のページの [コードソース] セクション (**tutorial-lambda-convert** など) で、[アップロード元]、[.zip ファイル]の順に選択します。
2. アップロード を選択して、ローカルの `.zip` ファイルを選択します。
3. 前に作成した `lambda.zip` ファイルを選択し、開く を選択します。
4. [Save] を選択します。
5. ランタイム設定 セクションで、編集 を選択します。
6. Lambda ランタイムに Lambda 関数コード内のどのハンドラメソッドの呼び出しを指示するには、ハンドラー フィールドに **convert.handler** を入力します。

   Python で関数を設定するとき、ハンドラー設定の値は、ファイル名と、ハンドラーモジュールの名前を、ドット (.) で区切ったものになります。例えば、`convert.handler` は、`convert.py` ファイルで定義された `handler` メソッドを呼び出します。
7. [Save] を選択します。
8. [Lambda 関数] ページで、設定 タブを選択します。設定 タブの左側のナビゲーションペインで、環境変数 を選択し、編集 を選択します。
9. 環境変数の追加 を選択します。次に、次の環境変数ごとにキー および値 を入力します。

   - キー : **DestinationBucket** 値: **tutorial-bucket-2**

この値は、ステップ 1 で作成した出力メディアファイルの S3 バケットです。

- キー : **MediaConvertRole** 値: **arn:aws:iam::*111122223333*:role/tutorial-mediaconvert-role**

  この値は、ステップ 2 で作成した MediaConvert の IAM ロールの ARN です。この ARN を IAM ロールの実際の ARN に置き換えてください。

- キー : **Application** 値: **Batch-Transcoding**

  この値は、アプリケーションの名前です。

10. [Save] を選択します。
11. (オプション) Configuration (設定) タブの左側のナビゲーションペインの General configuration (一般設定) セクションで、Edit (編集) を選択します。[タイムアウト] フィールドに、**2** 分 **0** 秒と入力します。次に、保存を選択します。

    タイムアウトは、Lambda で関数が停止するまでに許可される実行時間の長さです。デフォルト値は 3 秒です。料金は、設定されたメモリの量とコードの実行時間に基づいて請求されます。詳細については、AWS Lambda 料金表を参照してください。

ステップ 5: S3 ソースバケットの Amazon S3 インベントリを設定する。

トランスコーディング Lambda 関数をセットアップしたら、S3 バッチ操作ジョブを作成して、一連の動画をトランスコードします。まず、S3 バッチ操作で指定したトランスコーディングアクションを実行する入力動画オブジェクトのリストが必要です。入力動画オブジェクトのリストを取得するには、S3 ソースバケットの S3 インベントリレポートを生成できます (例、**tutorial-bucket-1**)。

サブステップ

- 入力動画の S3 インベントリレポート用のバケットを作成して設定する
- S3 動画ソースバケットの Amazon S3 インベントリを設定します。
- S3 動画ソースバケットのインベントリレポートを確認します。

入力動画の S3 インベントリレポート用のバケットを作成して設定する

S3 ソースバケットのオブジェクトをリストする S3 インベントリレポートを保存するには、S3 インベントリ宛先バケットを作成し、そのバケットに対してインベントリファイルを S3 ソースバケットに書き込むようにバケットポリシーを設定します。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [Create bucket (バケットの作成)] を選択します。
4. Bucket Name に、バケットの名前 (例: **tutorial-bucket-3**) を入力します。
5. AWS リージョン で、バケットを保存するAWS リージョン を選択します。

   インベントリ宛先バケットは、S3 インベントリを設定するソースバケットと同じ AWS リージョン である必要があります。インベントリ宛先バケットは、別の AWS アカウント にある場合があります。
6. このバケットのパブリックアクセス設定をブロック で、デフォルト設定 (ブロックすべてパブリックアクセスが有効) のままであることを確認します。
7. 残りの設定はデフォルトのままにしておきます。
8. [Create bucket (バケットの作成)] を選択します。
9. [バケット] リストで、作成したばかりのバケットの名前を選択します(例、**tutorial-bucket-3**) 。
10. S3 インベントリレポートのデータを S3 インベントリ送信先バケットに書き込む許可を Amazon S3 に付与するには、許可 タブを選択します。
11. バケットポリシー セクションまで下にスクロールし、編集 を選択します。バケットポリシーページが開きます。
12. S3 インベントリの許可を付与するには、ポリシー フィールドに、次のバケットポリシーを貼り付けます。

    3 つの例の値を、それぞれ次の値に置き換えます。

    - インベントリレポート (例、*tutorial-bucket-3*) を保存するために作成したバケットの名前。
    - 入力動画を保存するソースバケットの名前 (例、*tutorial-bucket-1*)。
    - S3 動画ソースバケットの作成に使用した AWS アカウント ID (例、*111122223333*)。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"InventoryAndAnalyticsExamplePolicy",
```

```
      "Effect":"Allow",
      "Principal": {"Service": "s3.amazonaws.com"},
      "Action":"s3:PutObject",
      "Resource":["arn:aws:s3:::tutorial-bucket-3/*"],
      "Condition": {
          "ArnLike": {
              "aws:SourceArn": "arn:aws:s3:::tutorial-bucket-1"
           },
          "StringEquals": {
              "aws:SourceAccount": "111122223333",
              "s3:x-amz-acl": "bucket-owner-full-control"
           }
       }
    }
  ]
}
```

13. [Save changes] (変更の保存) をクリックします。

S3 動画ソースバケットの Amazon S3 インベントリを設定します。

動画オブジェクトとメタデータのフラットファイルリストを生成するには、S3 動画ソースバケットの S3 インベントリを設定する必要があります。これらのスケジュールされたレポートには、バケット内のすべてのオブジェクト、または共有プレフィックスでグループ化されたオブジェクトを含めることができます。このチュートリアルでは、S3 インベントリレポートに S3 ソースバケット内のすべての動画オブジェクトが含まれます。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. S3 ソースバケットの入力動画の S3 インベントリレポートを設定するには、バケット リストで、S3 ソースバケットの名前を選択します (例、**tutorial-bucket-1**)。
4. [管理] タブを選択します。
5. インベントリの設定 セクションを下にスクロールし、インベントリ設定を作成する を選択します。
6. Inventory configuration name (インベントリ設定名) に、名前を入力します (例、**tutorial-inventory-config**)。
7. インベントリの範囲 で、オブジェクトバージョン に対して現在のバージョンのみ を選択し、他のインベントリの範囲 設定をこのチュートリアルのデフォルトに設定したままにします。

8. レポートの詳細 の 宛先バケット で、このアカウント を選択します。
9. 宛先 で S3 を参照 を選択し、インベントリレポートを保存するために先に作成した宛先バケットを選択します (例、**tutorial-bucket-3**)。次に、パスの選択 を選択します。

   インベントリ宛先バケットは、S3 インベントリを設定するソースバケットと同じ AWS リージョン である必要があります。宛先バケットは、別の AWS アカウント アカウントにある場合があります。

   [送信先] バケットフィールドの下に、Amazon S3 がそのバケットにデータを入れることを許可するために送信先バケットポリシーに追加される [送信先バケットの許可] が追加されます。詳細については、「ターゲットバケットポリシーの作成」を参照してください。
10. 頻度 で、1 日 1 回 を選択します。
11. [Output format] (出力形式) として [CSV] を選択します。
12. Status (ステータス) で、有効 を選択します。
13. サーバー側の暗号化 で、このチュートトリアルの 無効 を選択します。

    詳細については、S3 コンソールを使用したインベントリの設定およびカスタマーマネージドキーを暗号化に使用するためのアクセス許可を Amazon S3 に付与するを参照してください。
14. 追加フィールド - オプション セクションで、サイズ、最終更新日時 および ストレージクラス を選択します。
15. [Create] (作成) を選択します。

詳細については、「S3 コンソールを使用したインベントリの設定」を参照してください。

S3 動画ソースバケットのインベントリレポートを確認します。

インベントリレポートが発行されると、マニフェストファイルは S3 インベントリ宛先バケットに送信されます。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. Buckets (バケット) リストで、動画送信先バケットの名前を選択します (例、**tutorial-bucket-1**)。
4. [Management (管理)] を選択します。

5. S3 インベントリレポートで S3 バッチ操作ジョブを作成する準備ができているかどうかを確認するには、ステップ 7 の インベントリ設定 で、マニフェストからジョブを作成 ボタンが有効になっているかを確認します。

   > **Note**
   >
   > 最初のレポートが配信されるまでに最大で 48 時間かかることがあります。Create job from manifest (マニフェストからジョブを作成) ボタンが無効になっている場合、最初のインベントリレポートは配信されていません。ステップ 7 で S3 バッチ操作ジョブを作成するには、最初のインベントリレポートが配信され、マニフェストからジョブを作成 ボタンが有効になるまで待つ必要があります。

6. S3 インベントリレポート (`manifest.json`) を確認するには、宛先 列で、インベントリレポートを保存するために先に作成したインベントリ宛先バケットの名前を選択します (例、**`tutorial-bucket-3`**)。
7. オブジェクト タブで、S3 ソースバケットの名前を持つ既存のフォルダ (例、**`tutorial-bucket-1`**) を選択します。次に、先にインベントリ設定を作成した際に インベントリの設定名 で入力した名前を選択します (例、**`tutorial-inventory-config`**)。

   レポートの生成日を名前として含むフォルダのリストを表示できます。
8. 特定の日付の日次 S3 インベントリレポートを確認するには、生成日の名前が付いたフォルダを選択して、`manifest.json` を選択します。
9. 特定の日付のインベントリレポートの詳細を確認するには、manifest.json ページで、Download (ダウンロード) または Open (開く) を選択します。

ステップ 6: S3 バッチ操作の IAM ロールを作成する。

S3 バッチ操作を使用してバッチトランスコーディングを行うには、最初に IAM ロールを作成して、S3 バッチ操作を実行する許可を Amazon S3 に付与する必要があります。

サブステップ

- S3 バッチ操作用の IAM ポリシーを作成する。
- S3 バッチ操作 IAM ロールを作成し、許可ポリシーを割り当てる

S3 バッチ操作用の IAM ポリシーを作成する。

S3 バッチ操作に、入力マニフェストを読み取り、Lambda 関数を呼び出し、S3 バッチ操作ジョブ完了レポートに書き込みを行う許可を与える IAM ポリシーを作成する必要があります。

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインの [ポリシー] を選択します。
3. [Create policy] を選択します。
4. [JSON] タブを選択します。
5. JSON テキストフィールドに、次の JSON ポリシーをコピーして貼り付けます。

   JSON ポリシーで、4 つのサンプル値を次の値に置き換えます。

   - 入力動画を保存するソースバケットの名前 (例、*tutorial-bucket-1*)。
   - `manifest.json` ファイル (例、*tutorial-bucket-3*) を保存するためにステップ 5 で作成したインベントリ送信先バケットの名前。
   - 出力メディアファイルを保存するために ステップ 1 で作成したバケットの名前 (例、*tutorial-bucket-2*)。このチュートリアルでは、出力メディアファイルの送信先バケットにジョブ完了レポートを配置します。
   - ステップ 4 で作成した Lambda 関数のロール ARN。Lambda 関数のロール ARN を検索してコピーするには、次の操作を行います。
     - 新しいブラウザタブで、https://console.aws.amazon.com/lambda/home#/functions にある Lambda コンソールの 関数 ページを開きます。
     - 関数 のリストから、ステップ 4 で作成した Lambda 関数を選択します (例、**tutorial-lambda-convert**)。
     - [ARN のコピー] を選択します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Get",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
```

```
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::tutorial-bucket-1/*",
                "arn:aws:s3:::tutorial-bucket-3/*"
            ]
        },
        {
            "Sid": "S3PutJobCompletionReport",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::tutorial-bucket-2/*"
        },
        {
            "Sid": "S3BatchOperationsInvokeLambda",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-west-2:111122223333:function:tutorial-lambda-
convert"
            ]
        }
    ]
}
```

6. [Next: Tags] (次へ: タグ) を選択します。
7. [次へ: レビュー] を選択します。
8. [Name (名前)] フィールドに **tutorial-s3batch-policy** を入力します。
9. [Create policy] を選択します。

S3 バッチ操作 IAM ロールを作成し、許可ポリシーを割り当てる

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左側のナビゲーションペインから、[Roles (ロール)] を選択し、[Create role (ロールの作成)] を選択します。
3. AWS のサービス ロールタイプを選択し、S3 サービスを選択します。
4. [ユースケースを選択] で、S3 バッチ操作 を選択します。

5. [Next: Permissions] (次へ: アクセス許可) を選択します。
6. 許可ポリシーをアタッチする で、事前に作成した IAM ポリシーの名前 (例、**tutorial-s3batch-policy**) を検索ボックスに入力し、ポリシーのリストをフィルタリングします。ポリシー名の横にあるチェックボックスを選択します (例、**tutorial-s3batch-policy**)。
7. [Next: Tags] (次へ: タグ) を選択します。
8. [次へ: レビュー] を選択します。
9. [ロール名] に「**tutorial-s3batch-role**」と入力します。
10. [ロールの作成] を選択します。

    S3 バッチ操作の IAM ロールを作成した後は、次の信頼ポリシーが自動的にロールに添付されます。この信頼ポリシーは、S3 バッチ操作のサービスプリンシパルが IAM ロールを引き受けることを許可します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

ステップ 7: S3 バッチ操作ジョブを作成して実行する。

S3 バッチ操作 ジョブを作成して S3 ソースバケット内の入力動画を処理するには、この特定のジョブのパラメータを指定する必要があります。

> **Note**
>
> S3 バッチ操作ジョブの作成を開始するには、[マニフェストからジョブを作成] ボタンが有効になっていることを確認する必要があります。詳細については、「S3 動画ソースバケットのインベントリレポートを確認します。」を参照してください。マニフェストからジョブを作成 ボタンが無効になっている場合、最初のインベントリレポートは配信されておらず、ボタンが有効になるまで待つ必要があります。ステップ 5 で S3 ソースバケットの Amazon S3

インベントリを設定した後は、最初のインベントリレポートが配信されるまでに最大で 48 時間かかることがあります。

サブステップ

- S3 バッチオペレーションジョブを作成します。
- S3 バッチ操作ジョブを実行して Lambda 関数を呼び出します。
- (オプション) 完了レポートを確認する。
- (オプション) Lambda コンソールで各 Lambda 呼び出しをモニタリングする。
- (オプション)MediaConvert コンソールで各 MediaConvert 動画トランスコーディングジョブを監視する

S3 バッチオペレーションジョブを作成します。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[バッチ操作] を選択します。
3. [ジョブの作成]を選択します。
4. AWS リージョン については、ジョブを作成する [Region (リージョン)] を選択します。

   このチュートリアルで、S3 バッチ操作ジョブを使用して Lambda 関数を呼び出すには、マニフェストで参照されるオブジェクトがある S3 動画ソースバケットと同じリージョンにジョブを作成する必要があります。
5. マニフェスト セクションで、以下を実行します。

   a. Manifest format (マニフェスト形式) で、S3 inventory report (manifest.json) を選択します。

   b. オブジェクトのマニフェスト については、Browse S3 を選択して、インベントリレポートを保存するために ステップ 5 で作成したバケットを見つけます (例、**tutorial-bucket-3**)。マニフェストオブジェクト ページで、特定の日付の `manifest.json` ファイルが見つかるまで、オブジェクト名を検索します。このファイルには、バッチトランスコードするすべての動画に関する情報がリストされます。使用する `manifest.json` ファイルを見つけたら、横にあるオプションボタンを選択します。次に、[パスの選択] を選択します。

   c. (オプション) [マニフェストオブジェクトのバージョン ID-オプション] で、最新版以外のバージョンを使用する場合は、マニフェストオブジェクトのバージョン ID を入力します。

6. [Next] を選択します。
7. Lambda 関数を使用して、選択した `manifest.json` ファイルリストされているすべてのオブジェクトをトランスコードするには、オペレーションタイプ で AWS Lambda 関数を呼び出すを選択します。
8. Lambda 関数の呼び出し ウィンドウで、次を実行します。
   a. Choose from functions in your account (アカウントの関数を選択する) を選択します。
   b. Lambda 関数 で、ステップ 4 で作成した Lambda 関数を選択します (例、**`tutorial-lambda-convert`**)。
   c. Lambda 関数のバージョン で、デフォルト値の $LATEST を維持します。
9. [Next] を選択します。追加のオプションの設定 ページが開きます。
10. 追加のオプション で、デフォルトの設定を維持します。

    これらのパラメータの詳細については、「バッチ操作ジョブのリクエストの要素」を参照してください。
11. 完了レポート セクションの 完了レポートの送信先へのパス で、S3 をブラウズ を選択します。ステップ 1 で出力メディアファイル用に作成したバケットを見つけます (例、**`tutorial-bucket-2`**)。バケット名の横にあるオプションボタンを選択します。次に、パスの選択 を選択します。

    残りの 完了レポート 設定は、デフォルト値のままにしておきます。レポートの設定の完了に関する詳細については、「バッチ操作ジョブのリクエストの要素」を参照してください。完了レポートには、ジョブの詳細と実行された操作のレコードが保持されます。
12. 許可 で、既存の IAM ロールから選択する を選択します。IAM role (IAM ロール) で、ステップ 6 (例、**`tutorial-s3batch-role`**) で作成した S3 バッチ操作ジョブの IAM ロールを選択します。
13. [Next] を選択します。
14. Review ページで、設定を確認します。次に、ジョブの作成 を選択します。

    S3 が S3 バッチ操作のジョブのマニフェストの読み取りを終了すると、実行のための確認待ちの ステータス に移行します。ジョブのステータスの更新を表示するには、ページをリフレッシュします。ステータスが 実行のための確認待ち になるまで、ジョブを実行することはできません。

S3 バッチ操作ジョブを実行して Lambda 関数を呼び出します。

バッチ操作ジョブを実行して、動画トランスコーディング用の Lambda 関数を呼び出します。ジョブが失敗した場合は、完了レポートを確認して原因を特定できます。

S3 バッチ操作ジョブを実行するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、バッチ操作 を選択します。
3. ジョブ のリストから、最初の行のジョブの ジョブ ID を選択します。これは、上記で作成した S3 バッチ操作ジョブです。
4. [Run job] (ジョブの実行) を選択します。
5. ジョブパラメータをもう一度確認し、Total objects listed in manifest (マニフェストにリストされているオブジェクトの総数) の値がマニフェスト内のオブジェクトの数と同じであることを確認します。次に、ジョブの実行 を選択します。

   S3 バッチ操作ジョブページが開きます。
6. ジョブの実行が開始されたら、ジョブページの Status (ステータス) で、Status (ステータス)、% Complete (% 完了)、Total succeeded (rate) (成功合計) (レート)、Total failed (rate) (失敗合計) (レート)、Date terminated (終了日)、Reason for termination (終了理由) など、S3 バッチ操作ジョブの進行状況を確認します。

   S3 バッチ操作ジョブが完了したら、ジョブページのデータを表示して、ジョブが期待どおりに完了したことを確認します。

   1,000 を超える操作が試行された後、失敗が S3 バッチ操作ジョブのオブジェクト操作の 50% を超えると、ジョブは自動的に失敗します。完了レポートを確認して失敗の原因を特定するには、以下のオプション手順を参照してください。

(オプション) 完了レポートを確認する。

完了レポートを使用して、失敗したオブジェクトと失敗の原因を特定できます。

失敗したオブジェクトの詳細について完了レポートを確認するには

1. S3 バッチ操作ジョブのページで、完了レポート までスクロールし、完了レポートの宛先 のリンクを選択します。

S3 出力宛先バケットページが開きます。

2. オブジェクト タブで、上記で作成した S3 バッチ操作ジョブのジョブ ID で終わる名前のフォルダを選択します。
3. results/ (結果/) を選択します。
4. `.csv` ファイルの横にあるチェックボックスを選択します。
5. ジョブレポートを表示するには、[オープン] または [ダウンロード] を選択します。

(オプション) Lambda コンソールで各 Lambda 呼び出しをモニタリングする。

S3 バッチ操作ジョブの実行が開始されると、ジョブは入力動画オブジェクトごとに Lambda 関数の呼び出しを開始します。S3 では、各 Lambda 呼び出しのログを CloudWatch Logs に書き込みます。Lambda コンソールのモニタリングダッシュボードを使用して、Lambda 関数とアプリケーションをモニタリングできます。

1. AWS Lambda コンソールを https://console.aws.amazon.com/lambda/ で開きます。
2. 左側のナビゲーションペインで、[関数] を選択します。
3. 関数 のリストから、ステップ 4 で作成した Lambda 関数 の名前を選択します (例、**`tutorial-lambda-convert`**)。
4. [Monitor] (モニタリング) タブを選択します。
5. Metrics (メトリクス) で、Lambda 関数のランタイムメトリクスを確認します。
6. Logs (ログ) で、CloudWatch Logs インサイトを使用して各 Lambda 呼び出しのログデータを表示します。

> **Note**
>
> Lambda 関数で S3 バッチ操作を使用する場合、Lambda 関数は各オブジェクトで呼び出されます。S3 バッチ操作ジョブが大きい場合、複数の Lambda 関数を同時に呼び出すことができ、Lambda の同時実行数が急増します。
> 各 AWS アカウント リージョンごとに Lambda 同時実行クォータがあります。詳細については、「AWS Lambda デベロッパーガイド の「AWS Lambda 関数のスケーリング」を参照してください。S3 バッチ操作で Lambda 関数を使用するベストプラクティスは、Lambda 関数自体に同時実行制限を設定することです。同時実行制限を設定することにより、ジョブが Lambda の同時実行数のほとんどを消費したり、アカウント内の

他の機能をスロットリングする可能性がなくなります。詳細については、AWS Lambda デベロッパーガイド の Lambda 予約済同時実行数の管理 を参照してください。

(オプション)MediaConvert コンソールで各 MediaConvert 動画トランスコーディングジョブを監視する

MediaConvert ジョブはメディアファイルの変換作業を行います。S3 バッチ操作ジョブが動画ごとに Lambda 関数を呼び出すと、各 Lambda 呼び出しによって、入力動画ごとに MediaConvert トランスコーディングジョブが作成されます。

1. AWS Management Console にサインインし、MediaConvert コンソール (https://console.aws.amazon.com/mediaconvert/) を開きます。
2. MediaConvert 入門者向けページが表示される場合は、[Get Started] を選択します。
3. ジョブのリストから、各行を表示して、各入力動画のトランスコーディングタスクを監視します。
4. チェックするジョブの行を特定し、ジョブ ID リンクを選択してジョブ詳細ページを開きます。
5. [Job 概要] ページの [出力] で、ブラウザでサポートされている内容に応じて、HLS、MP4、またはサムネイル出力のリンクを選択し、出力メディアファイルの S3 送信先バケットに移動します。
6. S3 出力先バケットの対応するフォルダ(HLS、MP4、またはサムネイル)で、出力メディアファイルオブジェクトの名前を選択します。

   オブジェクトの詳細ページが開きます。
7. [オブジェクトの概要] のオブジェクトの詳細ページで、オブジェクト URL のリンクを選択して、トランスコードされた出力メディアファイルを監視します。

ステップ 8: S3 宛先バケットから出力メディアファイルを確認する。

S3 宛先バケットから出力メディアファイルを確認するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. バケット のリストから、ステップ 1 で作成した出力メディアファイルの S3 宛先バケットの名前を選択します (例、**`tutorial-bucket-2`**)。

4. [オブジェクト] タブで、各入力動画には、入力動画の名前を持つフォルダがあります。各フォルダには、入力動画のトランスコードされた出力メディアファイルが含まれています。

   入力動画の出力メディアファイルを確認するには、次の操作を行います。

   a. 確認したい入力動画の名前が入ったフォルダを選択します。
   b. [Default/] フォルダを選択します。
   c. トランスコード形式 (HLS、MP4、またはこのチュートリアルではサムネイル) のフォルダを選択します。
   d. 出力メディアファイルの名前を選択します。
   e. トランスコードされたファイルを見るには、オブジェクト詳細ページで、オブジェクト URL の下のリンクを選択します。

      HLS 形式の出力メディアファイルは、短いセグメントに分割されます。これらの動画を再生するには、互換性のあるプレーヤーに `.m3u8` ファイルのオブジェクト URL を埋め込む必要があります。

ステップ 9: クリーンアップする。

S3 バッチ操作、Lambda、MediaConvert を学習演習としてのみ使用して動画をトランスコードした場合は、割り当てた AWS リ送信先を削除して、料金が発生しないようにします。

サブステップ

- S3 ソースバケットの S3 インベントリ設定を削除する。
- Lambda 関数を削除する
- CloudWatch Logs グループを削除する
- IAM ロールのインラインポリシーとともに IAM ロールを削除する
- カスタマー管理の IAM ポリシーを削除する。
- S3 バケットを空にする
- S3 バケットの削除

S3 ソースバケットの S3 インベントリ設定を削除する。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

2. 左側のナビゲーションペインで、[バケット] を選択します。
3. バケット] リストで、送信先バケットの名前を選択します (例、**tutorial-bucket-1**)。
4. [管理] タブを選択します。
5. [インベントリ設定] セクションで、ステップ 5 で作成したインベントリ設定の横のオプションボタンを選択します (例、**tutorial-inventory-config**)。
6. [削除]、[確認] の順に選択します。

Lambda 関数を削除する

1. AWS Lambda コンソールを https://console.aws.amazon.com/lambda/ で開きます。
2. 左側のナビゲーションペインで、[関数] を選択します。
3. ステップ 4 で作成した関数の横にあるチェックボックスを選択します (例、**tutorial-lambda-convert**)。
4. [アクション] を選択し、[削除] を選択します。
5. [関数の削除] ダイアログボックスで、[削除] を選択します。

CloudWatch Logs グループを削除する

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 左のナビゲーションペインで、[ログ] 、[ロググループ] の順に選択します。
3. ステップ 4 で作成した Lambda 関数で終わる名前を持つロググループの横にあるチェックボックスを選択します (例、**tutorial-lambda-convert**)。
4. [アクション] を選択してから、[ロググループの削除] を選択します。
5. [ロググループの削除] ダイアログボックスで、[削除] をクリックします。

IAM ロールのインラインポリシーとともに IAM ロールを削除する

ステップ 2、ステップ 3、およびステップ 6] で作成した IAM ロールを削除するには、次の作業を行います。

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

2. ナビゲーションペインで、[ロール] を選択し、削除するロール名の隣にあるチェックボックスを選択します。
3. ページの上部で、[削除] を選択します。
4. 確認ダイアログボックスで、プロンプトに基づいてテキスト入力フィールドに必要な応答を入力し、[削除] を選択します。

カスタマー管理の IAM ポリシーを削除する。

ステップ 6 で作成したカスタマー管理の IAM ポリシーを削除するには、次の作業を行います。

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインの [ポリシー] を選択します。
3. ステップ 6 で作成したポリシーの横にあるオプションボタンを選択します (例、**tutorial-s3batch-policy**)。検索ボックスを使用して、グループのリストをフィルタリングできます。
4. [アクション] を選択し、[削除] を選択します。
5. テキストフィールドにポリシーの名前を入力して、このポリシーを削除することを確認し、[削除] を選択します。

S3 バケットを空にする

前提条件 、ステップ 1、および ステップ 5 で作成した S3 バケットを空にするには、次の作業を行います。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット名] リストで、空にするバケットの名前の横にあるバケットアイコンを選択し、[バケットを空にする] を選択します。
4. [バケットを空にする] ページで、テキストフィールドに **permanently delete** を入力することでバケットを空にすることを確定し、[空にする] を選択します。

S3 バケットの削除

前提条件 、ステップ 1 およびステップ 5 で作成した S3 バケットを削除するには、次の作業を行います。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、削除するバケット名の横にあるオプションボタンを選択します。
4. [削除] を選択します。
5. [バケットの削除] ページで、テキストフィールドにバケット名を入力することでバケットを削除することを確認し、[バケットを削除] を選択します。

次のステップ

このチュートリアルを完了すると、他の関連するユースケースをさらに詳しく調べることができます。

- Amazon CloudFront を使用して、トランスコードされたメディアファイルを世界中のビューワーにストリーミングできます。詳細については、「チュートリアル: Amazon S3、Amazon CloudFront、Amazon Route 53 を使用したオンデマンドストリーミング動画のホスティング。」を参照してください。
- S3 ソースバケットにアップロードした時点で、動画をトランスコードできます。これを行うには、MediaConvert を使用して S3 の新しいオブジェクトをトランスコードするために、Lambda 関数を自動的に呼び出す Amazon S3 イベントトリガーを設定できます。詳細については、AWS Lambdaデベロッパーガイド のチュートリアル: Amazon S3 トリガーを使用して Lambda 関数を呼び出す を参照してください。

チュートリアル: Amazon S3 での静的ウェブサイトの設定

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオ

ブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

Amazon S3 バケットは、ウェブサイトと同じように機能するように設定できます。この例では、Amazon S3 上でウェブサイトをホスティングするステップを説明します。

Important

次のチュートリアルでは、[パブリックアクセスをブロック] を無効にする必要があります。[パブリックアクセスをブロック] 設定は、有効にしておくことをお勧めします。4 つすべての [パブリックアクセスをブロック] 設定を有効にしたまま、静的ウェブサイトをホストする場合は、Amazon CloudFront のオリジンアクセスコントロール (OAC) を使用できます。Amazon CloudFront は、セキュアな静的ウェブサイトをセットアップするために必要な機能を提供します。Amazon S3 静的ウェブサイトは、HTTP エンドポイントのみをサポートしています。Amazon CloudFront は、耐久性に優れた Amazon S3 のあるストレージを使用し、HTTPS などの、追加のセキュリティヘッダーを提供します。HTTPS では、通常の HTTP リクエストを暗号化し、一般的なサイバー攻撃から保護することで、セキュリティが強化されます。詳細については、「Amazon CloudFront デベロッパーガイド」の「安全な静的ウェブサイトの使用開始」を参照してください。

トピック

ステップ 1: バケットを作成する

以下の手順では、ウェブサイトホスティングにバケットを作成する方法の概要を説明します。バケットの作成に関するステップバイステップの方法は、「バケットの作成」を参照してください。

バケットを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Create bucket (バケットの作成)] を選択します。
3. [バケット名] (**`example.com`** など) を入力します。
4. バケットを作成するリージョンを選択します。

   レイテンシーとコストを最小限に抑えるため、または規制要件に対処するために、最寄りのリージョンを選択します。選択したリージョンによって、Amazon S3 ウェブサイトエンドポイントが決まります。詳細については、「ウェブサイトエンドポイント」を参照してください。
5. デフォルト設定をそのまま使用してバケットを作成するには、[作成] を選択します。

ステップ 2: 静的ウェブサイトホスティングを有効にする

バケットを作成したら、バケットの静的ウェブサイトホスティングを有効にできます。新しいバケットを作成することも、既存のバケットを使用することもできます。

静的ウェブサイトホスティングを有効にするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、静的ウェブサイトホスティングを有効にするバケットの名前を選択します。
3. [プロパティ] を選択します。
4. [静的ウェブサイトホスティング] で [編集] を選択します。
5. [このバケットを使用してウェブサイトをホストする] を選択します。
6. [静的ウェブサイトホスティング] で [有効化] を選択します。
7. [インデックスドキュメント] に、インデックスドキュメントのファイル名 (通常は `index.html`) を入力します。

インデックスドキュメント名の大文字と小文字は区別されます。この名前は、S3 バケットにアップロードする HTML インデックスドキュメントのファイル名と正確に一致する必要があります。バケットをウェブサイトホスティング用に設定するときは、インデックスドキュメントを指定する必要があります。Amazon S3 からこのインデックスドキュメントが返されるのは、ルートドメインまたはサブフォルダに対するリクエストが行われたときです。詳細については、「インデックスドキュメントの設定」を参照してください。

8. 4XX クラスエラーに関する独自のカスタムエラードキュメントを指定する場合は、[エラードキュメント] にカスタムエラードキュメントのファイル名を入力します。

   エラードキュメント名の大文字と小文字は区別されます。この名前は、S3 バケットにアップロードする HTML エラードキュメントのファイル名と正確に一致する必要があります。カスタムエラードキュメントを指定しない場合、エラーが発生すると、Amazon S3 からデフォルトの HTML エラードキュメントが返されます。詳細については、「カスタムエラードキュメントの設定」を参照してください。

9. (オプション) 高度なリダイレクトツールを指定する場合は、[Rredirection rules] (リダイレクトルール) に、JSON を入力してルールを記述します。

   例えば、条件に応じてリクエストのルーティング先を変えることができます。この条件として使用できるのは、リクエストの中の特定のオブジェクトキー名またはプレフィックスです。詳細については、「高度な条件付きリダイレクトを使用するようにリダイレクトルールを設定する」を参照してください。

10. [Save changes] (変更の保存) をクリックします。

    Amazon S3 では、バケットの静的ウェブサイトホスティングを有効にします。ページの下部の [静的ウェブサイトホスティング] の下に、バケットのウェブサイトエンドポイントが表示されます。

11. [静的 ウェブサイトホスティング] の下のエンドポイントを書き留めます。

    [Endpoint (エンドポイント)] は、バケットの Amazon S3 ウェブサイトエンドポイントです。バケットを静的ウェブサイトとして設定すると、このエンドポイントを使用してウェブサイトをテストできます。

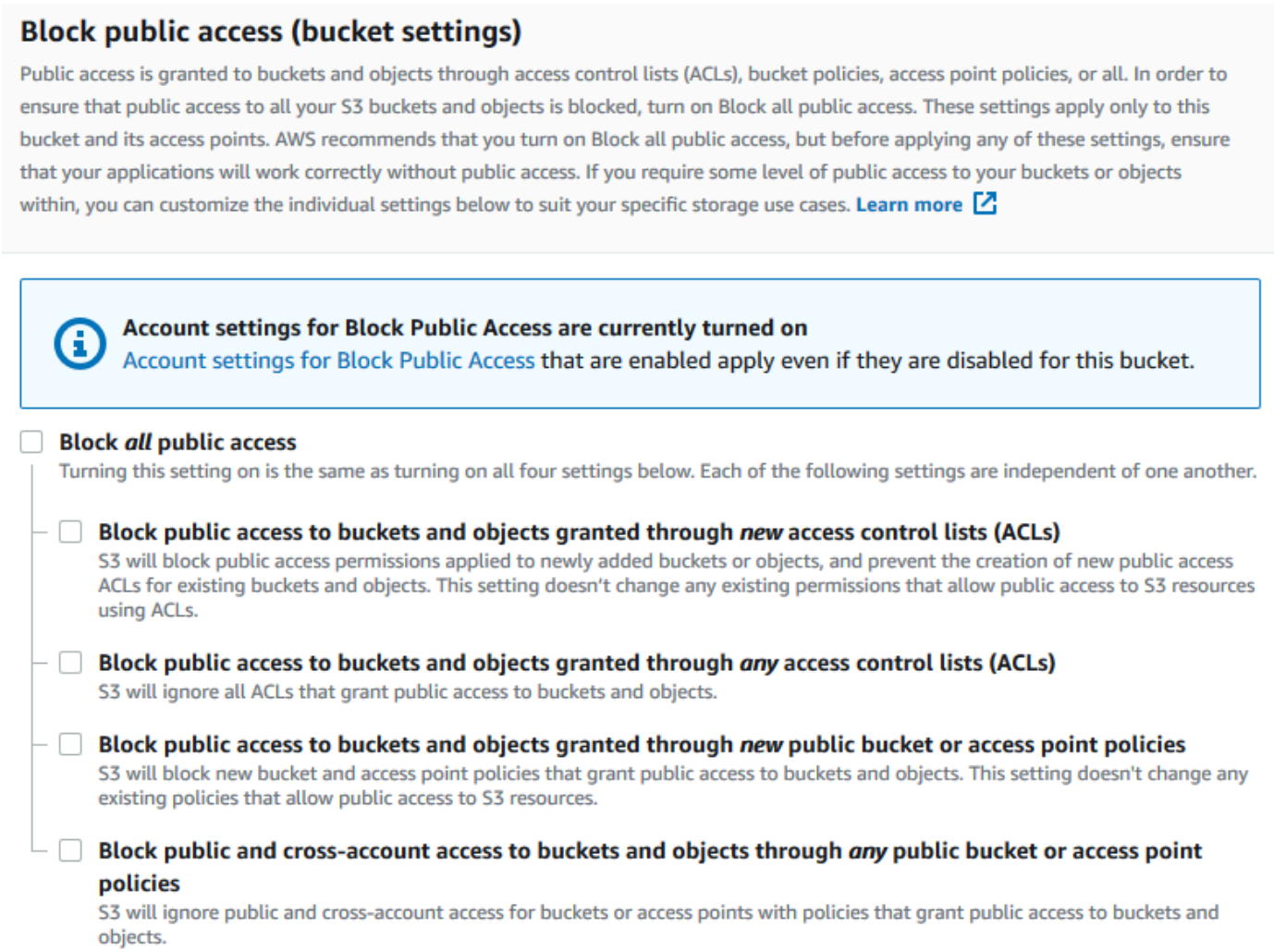
ステップ 3: パブリックアクセスブロック設定を編集する

デフォルトでは、Amazon S3 はアカウントとバケットへのパブリックアクセスをブロックします。バケットを使用して静的ウェブサイトをホストする場合は、以下のステップを使用して、パブリックアクセスブロック設定を編集できます。

> **Warning**
>
> このステップを完了する前に「Amazon S3 ストレージへのパブリックアクセスのブロック」を読んで、パブリックアクセスを許可することに伴うリスクを理解し、了承します。パブリックアクセスブロック設定をオフにしてバケットをパブリックにすると、インターネット上のだれでもバケットにアクセスできるようになります。バケットへのすべてのパブリックアクセスをブロックすることをお勧めします。

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 静的ウェブサイトとして設定されたバケットの名前を選択します。
3. [Permissions (アクセス許可)] を選択します。
4. [ブロックパブリックアクセス (バケット設定)] で [編集] を選択します。
5. [Block all public access (すべてのパブリックアクセスをブロックする)] をクリアし、[Save changes (変更の保存)] を選択します。

> **Warning**
>
> このステップを完了する前に、「Amazon S3 ストレージへのパブリックアクセスのブロック」を確認し、パブリックアクセスの許可に伴うリスクを理解したうえで了承してください。パブリックアクセスブロック設定をオフにしてバケットをパブリックにすると、インターネット上のだれでもバケットにアクセスできるようになります。バケットへのすべてのパブリックアクセスをブロックすることをお勧めします。



Amazon S3 は、バケットのパブリックアクセスブロック設定をオフにします。パブリックで静的ウェブサイトを作成するには、バケットポリシーを追加する前に、アカウントのブロックパブリックアクセス設定を編集する必要があります。パブリックアクセスのブロックのアカウント設定が現在有効になっている場合は、[Block public access (bucket settings) (パブリックアクセスのブロック (バケット設定))] の下にメモが表示されます。

ステップ 4: バケットの内容の公開を許可するバケットポリシーを追加する

S3 のパブリックアクセスブロック設定を編集した後で、バケットへのパブリック読み取りアクセスを許可するバケットポリシーを追加できます。パブリック読み取りアクセスを許可すると、インターネット上のだれでもバケットにアクセスできるようになります。

> **Important**
>
> 次のポリシーは、単なる例として、バケットのコンテンツへのフルアクセスを許可します。このステップに進む前に、「Amazon S3 バケット内のファイルを保護するにはどうすればよいですか?」を確認して、S3 バケット内のファイルを保護するためのベストプラクティスと、パブリックアクセスの許可に伴うリスクを理解してください。

1. [バケット] で、バケットの名前を選択します。
2. [Permissions (アクセス許可)] を選択します。
3. [Bucket Policy (バケットポリシー)] で [編集] を選択します。
4. ウェブサイトのパブリック読み取りアクセスを許可するには、次のバケットポリシーをコピーし、[Bucket policy editor (バケットポリシーエディター)] に貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. `Resource` をバケット名に更新します。

   前述のバケットポリシーの例では、*`Bucket-Name`* はバケット名のプレースホルダーです。このバケットポリシーを独自のバケットで使用するには、バケット名に合わせてこの名前を更新する必要があります。

6. [Save changes] (変更の保存) をクリックします。

   バケットポリシーが正常に追加されたことを示すメッセージが表示されます。

`Policy has invalid resource` というエラーが表示された場合は、バケットポリシー内のバケット名がバケット名と一致していることを確認します。バケットポリシーの追加については、「S3 バケットポリシーを追加する方法」を参照してください。

エラーメッセージが表示され、バケットポリシーを保存できない場合は、アカウントとバケットの [パブリックアクセスをブロックする] 設定をチェックして、バケットへのパブリックアクセスを許可していることを確認します。

ステップ 5: インデックスドキュメントを設定する

バケットに対して静的ウェブサイトホスティングを有効にする場合は、インデックスドキュメントの名前 (**`index.html`** など) を入力します。バケットに対して静的ウェブサイトホスティングを有効にした後、インデックスドキュメント名を含む HTML ファイルをバケットにアップロードします。

インデックスドキュメントを設定するには

1. `index.html` ファイルを作成します。

   `index.html` ファイルがない場合は、以下の HTML を使用して作成できます。

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>My Website Home Page</title>
</head>
<body>
  <h1>Welcome to my website</h1>
  <p>Now hosted on Amazon S3!</p>
</body>
</html>
```

2. インデックスファイルをローカルに保存します。

   インデックスドキュメントファイル名は、[静的ウェブサイトホスティング] ダイアログボックスで入力したインデックスドキュメント名と正確に一致する必要があります。インデックスドキュメント名では、大文字と小文字が区別されます。例えば、[静的ウェブサイトホスティング] ダイアログボックスの [インデックスドキュメント] 名に「`index.html`」と入力する場合、インデックスドキュメントファイル名も `index.html` ではなく `Index.html` である必要があります。

3. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
4. [バケット] リストで、静的ウェブサイトホスティングに使用するバケットの名前を選択します。
5. バケットに対して静的ウェブサイトホスティングを有効にし、インデックスドキュメントの正確な名前 (`index.html` など) を入力します。詳細については、「ウェブサイトのホスティングの有効化」を参照してください。

   静的ウェブサイトホスティングを有効にしたら、ステップ 6 に進みます。
6. インデックスドキュメントをバケットにアップロードするには、以下のいずれかを実行します。

   - インデックスファイルをコンソールバケットのリストにドラッグアンドドロップします。
   - [Upload] を選択し、プロンプトに従ってインデックスファイルを選択してアップロードします。

   手順については、「オブジェクトのアップロード」を参照してください。
7. (オプション) 他のウェブサイトコンテンツをバケットにアップロードします。

ステップ 6: エラードキュメントの設定

バケットに対して静的ウェブサイトホスティングを有効にするときは、エラードキュメントの名前 (例: **`404.html`**) を入力します。バケットに対して静的ウェブサイトホスティングを有効にしたら、エラードキュメント名を含む HTML ファイルをバケットにアップロードします。

エラードキュメントを設定するには

1. エラードキュメント (例: `404.html`) を作成します。
2. エラードキュメントのファイルをローカルに保存します。

   エラードキュメントの名前は、大文字と小文字を区別し、静的ウェブサイトホスティングを有効にする際に入力した名前と厳密に一致している必要があります。たとえば、[Static website hosting] (静的ウェブサイトホスティング) ダイアログボックスの [Error document] (エラードキュメント) 名に `404.html` と入力する場合、エラードキュメントのファイル名も `404.html` である必要があります。
3. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
4. [バケット] リストで、静的ウェブサイトホスティングに使用するバケットの名前を選択します。

5. バケットに対して静的ウェブサイトホスティングを有効にし、エラードキュメントの正確な名前 (例: `404.html`) を入力します。詳細については、ウェブサイトのホスティングの有効化およびカスタムエラードキュメントの設定を参照してください。

   静的ウェブサイトホスティングを有効にしたら、ステップ 6 に進みます。

6. エラードキュメントをバケットにアップロードするには、以下のいずれかを実行します。

   - エラードキュメントファイルをコンソールバケットのリストにドラッグアンドドロップします。
   - [Upload] を選択し、プロンプトに従ってインデックスファイルを選択してアップロードします。

   手順については、「オブジェクトのアップロード」を参照してください。

ステップ 7: ウェブサイトエンドポイントをテストする

バケットに静的ウェブサイトホスティングを設定したら、ウェブサイトエンドポイントをテストできます。

> **Note**
>
> Amazon S3 は、ウェブサイトへの HTTPS アクセスをサポートしていません。HTTPS を使用する場合は、Amazon CloudFront を使用して Amazon S3 でホストされている静的ウェブサイトを提供できます。
> 詳細については、「CloudFront を使用して、Amazon S3 でホストされた静的ウェブサイトを公開するにはどうすればよいですか?」と「ビューワーと CloudFront との通信で HTTPS を必須にする」を参照してください。

1. [バケット] で、バケットの名前を選択します。
2. [プロパティ] を選択します。
3. ページの下部の [静的ウェブサイトホスティング] で、[Bucket website endpoint (バケットウェブサイトエンドポイント)] を選択します。

   インデックスドキュメントが別のブラウザウィンドウで開きます。

これで、Amazon S3 でウェブサイトがホスティングされるようになりました。このウェブサイトには、Amazon S3 ウェブサイトエンドポイントの URL を指定してアクセスできます。ただし、作成したウェブサイトのコンテンツを配信するのに、example.com などのドメインを使用することもできます。また、Amazon S3 のルートドメインサポートを利用すると、http://www.example.com と http://example.com のどちらのリクエストでも処理できるようになります。このようにするには、追加のステップが必要です。例については、「チュートリアル: Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定」を参照してください。

ステップ 8: クリーンアップする

学習のためだけに静的ウェブサイトを作成した場合は、割り当てた AWS リソースを削除して、料金が発生しないようにします。AWS リソースを削除すると、ウェブサイトは使用できなくなります。詳細については、「バケットの削除」を参照してください。

チュートリアル: Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定

Amazon S3 で静的ウェブサイトをホスティングするとします。ドメインは Amazon Route 53 で登録済みであり (例: example.com)、http://www.example.com と http://example.com へのリクエストに対しては Amazon S3 からコンテンツを配信するようにします。このチュートリアルでは、静的ウェブサイトをホスティングし、Route 53 に登録されているカスタムドメイン名を持つウェブサイトのリダイレクトを Amazon S3 で作成する方法について説明します。Amazon S3 でホストしようとしている既存のウェブサイトを使ってもよいですし、このチュートリアルで一から始めることもできます。

チュートリアルが完了したら、Amazon CloudFront を使ってウェブサイトのパフォーマンスを向上させることも可能です。詳細については、「Amazon CloudFront によるウェブサイトの高速化」を参照してください。

> **Note**
>
> Amazon S3 ウェブサイトエンドポイントは HTTPS またはアクセスポイントをサポートしていません。HTTPS を使用する場合は、Amazon CloudFront を使用して Amazon S3 でホストされている静的ウェブサイトを提供できます。
> CloudFront と Amazon S3 でコンテンツを安全にホストする方法のチュートリアルについては、「チュートリアル: Amazon S3、Amazon CloudFront、Amazon Route 53 を使用したオンデマンドストリーミング動画のホスティング。」を参照してください。詳細については、

「CloudFront を使用して、Amazon S3 でホストされた静的ウェブサイトを公開するにはどうすればよいですか?」と「ビューワーと CloudFront との通信で HTTPS を必須にする」を参照してください。

AWS CloudFormation テンプレートを使用した静的ウェブサイトの設定の自動化

AWS CloudFormation テンプレートを使用すると、静的ウェブサイトの設定を自動化することができます。AWS CloudFormation テンプレートによって安全な静的ウェブサイトをホストするために必要なコンポーネントが設定されるため、コンポーネントの設定よりもウェブサイトのコンテンツに注力できるようになります。

AWS CloudFormation テンプレートには、以下のコンポーネントが含まれています。

- Amazon S3 - 静的ウェブサイトをホストする Amazon S3 バケットを作成します。
- CloudFront - 静的ウェブサイトを高速化するために、CloudFront ディストリビューションを作成します。
- Lambda@Edge - Lambda@Edge を使用して、サーバーのすべてのレスポンスにセキュリティヘッダーを追加します。セキュリティヘッダーは、ウェブサーバーレスポンス内のヘッダーのグループであり、追加のセキュリティ対策を講じるようにウェブブラウザに指示します。詳細については、ブログ記事「Adding HTTP security headers using Lambda@Edge and Amazon CloudFront」を参照してください。

この AWS CloudFormation テンプレートは、ダウンロードして使用することができます。詳細および手順については、Amazon CloudFront デベロッパーガイドの「安全な静的ウェブサイトの使用開始」を参照してください。

トピック

開始する前に

この例で実行するステップでは、以下のサービスを使用します。

Amazon Route 53 - Route 53 では、ドメインを登録し、ドメインのインターネットトラフィックをルーティングする先を定義します。以下の例では、Route 53 エイリアスレコードを作成して、ドメイン (`example.com`) とサブドメイン (`www.example.com`) のトラフィックを HTML ファイルが含まれている Amazon S3 バケットにルーティングする方法を示します。

Amazon S3 - Amazon S3 を使用して、バケットを作成したり、サンプルウェブサイトページをアップロードしたり、全員がコンテンツを表示できるようアクセス許可を設定したりします。また、ウェブサイトホスティング用にバケットを設定することもできます。

ステップ 1: カスタムドメインを Route 53 に登録する

登録済みドメイン名 (`example.com` など) がない場合は、Route 53 でドメイン名を登録します。詳細については、「Amazon Route 53 デベロッパーガイド」の「新しいドメインの登録」を参照してください。ドメイン名を登録したら、Amazon S3 バケットを作成してウェブサイトのホスティング用に設定できます。

ステップ 2: バケットを 2 つ作成する

ルートドメインとサブドメインからの両方のリクエストをサポートするには、2 つのバケットを作成します。

- ドメインバケット - `example.com`
- サブドメインバケット - `www.example.com`

これらのバケット名はドメイン名と厳密に一致している必要があります。この例では、ドメイン名は example.com です。コンテンツをルートドメインバケット (example.com) からホストします。サブドメインバケットのリダイレクトリクエストを作成します (www.example.com)。他のユーザーがブラウザに www.example.com を入力すると、example.com にリダイレクトされ、その名前で Amazon S3 バケットにホストされているコンテンツが表示されます。

バケットをウェブサイトホスティング用に作成するには

以下の手順では、ウェブサイトホスティングにバケットを作成する方法の概要を説明します。バケットの作成に関するステップバイステップの方法は、「バケットの作成」を参照してください。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ルートドメインのバケットを作成します。
   a. [Create bucket (バケットの作成)] を選択します。
   b. [バケット名] (**example.com** など) を入力します。
   c. バケットを作成するリージョンを選択します。

      レイテンシーとコストを最小限に抑えるため、または規制要件に対処するために、最寄りのリージョンを選択します。選択したリージョンによって、Amazon S3 ウェブサイトエンドポイントが決まります。詳細については、「ウェブサイトエンドポイント」を参照してください。
   d. デフォルト設定をそのまま使用してバケットを作成するには、[作成] を選択します。
3. サブドメインのバケットを作成します。
   a. [Create bucket (バケットの作成)] を選択します。
   b. [バケット名] (**www.example.com** など) を入力します。
   c. バケットを作成するリージョンを選択します。

      レイテンシーとコストを最小限に抑えるため、または規制要件に対処するために、最寄りのリージョンを選択します。選択したリージョンによって、Amazon S3 ウェブサイトエンドポイントが決まります。詳細については、「ウェブサイトエンドポイント」を参照してください。
   d. デフォルト設定をそのまま使用してバケットを作成するには、[作成] を選択します。

次のステップでは、ウェブサイトホスティング用に example.com を設定します。

ステップ 3: ウェブサイトホスティング用にルートドメインのバケットを設定する

このステップでは、ルートドメインのバケット (example.com) をウェブサイトとして設定します。このバケットには、ウェブサイトのコンテンツが含まれます。バケットをウェブサイトホスティング用に設定すると、そのウェブサイトには ウェブサイトエンドポイント を使用してアクセスできるようになります。

静的ウェブサイトホスティングを有効にするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、静的ウェブサイトホスティングを有効にするバケットの名前を選択します。
3. [プロパティ] を選択します。
4. [静的ウェブサイトホスティング] で [編集] を選択します。
5. [このバケットを使用してウェブサイトをホストする] を選択します。
6. [静的ウェブサイトホスティング] で [有効化] を選択します。
7. [インデックスドキュメント] に、インデックスドキュメントのファイル名 (通常は index.html) を入力します。

   インデックスドキュメント名の大文字と小文字は区別されます。この名前は、S3 バケットにアップロードする HTML インデックスドキュメントのファイル名と正確に一致する必要があります。バケットをウェブサイトホスティング用に設定するときは、インデックスドキュメントを指定する必要があります。Amazon S3 からこのインデックスドキュメントが返されるのは、ルートドメインまたはサブフォルダに対するリクエストが行われたときです。詳細については、「インデックスドキュメントの設定」を参照してください。
8. 4XX クラスエラーに関する独自のカスタムエラードキュメントを指定する場合は、[エラードキュメント] にカスタムエラードキュメントのファイル名を入力します。

   エラードキュメント名の大文字と小文字は区別されます。この名前は、S3 バケットにアップロードする HTML エラードキュメントのファイル名と正確に一致する必要があります。カスタムエラードキュメントを指定しない場合、エラーが発生すると、Amazon S3 からデフォルトの HTML エラードキュメントが返されます。詳細については、「カスタムエラードキュメントの設定」を参照してください。

9. (オプション) 高度なリダイレクトツールを指定する場合は、[Rredirection rules] (リダイレクトルール) に、JSON を入力してルールを記述します。

   例えば、条件に応じてリクエストのルーティング先を変えることができます。この条件として使用できるのは、リクエストの中の特定のオブジェクトキー名またはプレフィックスです。詳細については、「高度な条件付きリダイレクトを使用するようにリダイレクトルールを設定する」を参照してください。

10. [Save changes] (変更の保存) をクリックします。

    Amazon S3 では、バケットの静的ウェブサイトホスティングを有効にします。ページの下部の [静的ウェブサイトホスティング] の下に、バケットのウェブサイトエンドポイントが表示されます。

11. [静的 ウェブサイトホスティング] の下のエンドポイントを書き留めます。

    [Endpoint (エンドポイント)] は、バケットの Amazon S3 ウェブサイトエンドポイントです。バケットを静的ウェブサイトとして設定すると、このエンドポイントを使用してウェブサイトをテストできます。

パブリックアクセスのブロック設定を編集し、パブリック読み取りアクセスを許可するバケットポリシーを追加したら、ウェブサイトエンドポイントを使用してウェブサイトにアクセスできます。

次のステップで、リクエストをドメイン (`www.example.com`) にリダイレクトするように、サブドメイン (`example.com`) を設定します。

ステップ 4: ウェブサイトのリダイレクト用にサブドメインのバケットを設定する

ウェブサイトホスティング用にルートドメインのバケットを設定したら、このドメインにすべてのリクエストをリダイレクトするように、サブドメインのバケットを設定できます。この例では `www.example.com` のすべてのリクエストが `example.com` にリダイレクトされます。

リダイレクトリクエストを設定するには

1. Amazon S3 コンソールの [Buckets (バケット)] リストで、サブドメインのバケット (この例では `www.example.com`) を選択します。
2. [プロパティ] を選択します。
3. [静的ウェブサイトホスティング] で [編集] を選択します。

4. [Redirect requests for an object (オブジェクトのリクエストをリダイレクト)] を選択します。
5. [Target bucket (ターゲットバケット)] ボックスに、ルートドメイン (**example.com** など) を入力します。
6. [Protocol (プロトコル)] で、[http] を選択します。
7. [Save changes] (変更の保存) をクリックします。

ステップ 5: ウェブサイトトラフィックのログ記録を設定する

ウェブサイトにアクセスする閲覧者の数を追跡するには、オプションでルートドメインのバケットのログ記録を有効にできます。詳細については、「サーバーアクセスログによるリクエストのログ記録」を参照してください。Amazon CloudFront を使用してウェブサイトを高速化する場合は、CloudFront のログ記録を使用することもできます。

ルートドメインバケット用のサーバーアクセスのログ記録を有効にするには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 静的ウェブサイトとして設定しているバケットを作成したのと同じリージョンで、ログ記録用のバケットを作成します (例: `logs.example.com`)。
3. サーバアクセスのログを記録するログファイル用のフォルダを作成します (例: `logs`)。
4. (オプション) CloudFront を使用してウェブサイトのパフォーマンスを向上させる場合は、CloudFront ログファイル用のフォルダを作成します (例: `cdn`)。

> **Important**
>
> ディストリビューションを作成または更新して CloudFront のログを有効にすると、CloudFront はバケットのアクセスコントロールリスト (ACL) を更新して、バケットにログを書き込む `awslogsdelivery` アクセス許可を `FULL_CONTROL` アカウントに付与します。詳細については、「Amazon CloudFront デベロッパーガイド」の「標準ログ記録の設定とログファイルへのアクセスに必要なアクセス許可」を参照してください。ログを保存するバケットが S3 オブジェクト所有権のバケット所有者の強制設定を使用して ACL を無効にすると、CloudFront はバケットにログを書き込むことができません。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

5. [Bucket (バケット)] リストで、ルートドメインのバケットを選択します。
6. [プロパティ] を選択します。

7. [Server access logging (サーバーアクセスのログ記録)] で、[Edit (編集)] を選択します。
8. [Enable] を選択します。
9. [Target bucket (ターゲットバケット)] で、サーバーアクセスログのバケットとフォルダの宛先を選択します。

   - フォルダとバケットの場所を参照します。
     1. [Browse S3(S3 の参照)] を選択します。
     2. バケット名を選択し、ログフォルダを選択します。
     3. [Choose path (パスの選択)] を選択します。
   - S3 バケットパスを入力します (例: `s3://logs.example.com/logs/`)。
10. [Save changes] (変更の保存) をクリックします。

    ログバケットで、ログにアクセスできるようになりました。Amazon S3 は 2 時間おきにログバケットにウェブサイトアクセスログを書き込みます。

ステップ 6: インデックスとウェブサイトのコンテンツをアップロードする

このステップでは、インデックスドキュメントとオプションのウェブサイトコンテンツをルートドメインのバケットにアップロードします。

バケットに対して静的ウェブサイトホスティングを有効にする場合は、インデックスドキュメントの名前 (**`index.html`** など) を入力します。バケットに対して静的ウェブサイトホスティングを有効にした後、インデックスドキュメント名を含む HTML ファイルをバケットにアップロードします。

インデックスドキュメントを設定するには

1. `index.html` ファイルを作成します。

   `index.html` ファイルがない場合は、以下の HTML を使用して作成できます。

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>My Website Home Page</title>
</head>
<body>
  <h1>Welcome to my website</h1>
  <p>Now hosted on Amazon S3!</p>
</body>
```

```
</html>
```

2. インデックスファイルをローカルに保存します。

   インデックスドキュメントファイル名は、[静的ウェブサイトホスティング] ダイアログボックスで入力したインデックスドキュメント名と正確に一致する必要があります。インデックスドキュメント名では、大文字と小文字が区別されます。例えば、[静的ウェブサイトホスティング] ダイアログボックスの [インデックスドキュメント] 名に「index.html」と入力する場合、インデックスドキュメントファイル名も index.html ではなく Index.html である必要があります。
3. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
4. [バケット] リストで、静的ウェブサイトホスティングに使用するバケットの名前を選択します。
5. バケットに対して静的ウェブサイトホスティングを有効にし、インデックスドキュメントの正確な名前 (index.html など) を入力します。詳細については、「ウェブサイトのホスティングの有効化」を参照してください。

   静的ウェブサイトホスティングを有効にしたら、ステップ 6 に進みます。
6. インデックスドキュメントをバケットにアップロードするには、以下のいずれかを実行します。

   - インデックスファイルをコンソールバケットのリストにドラッグアンドドロップします。
   - [Upload] を選択し、プロンプトに従ってインデックスファイルを選択してアップロードします。

   手順については、「オブジェクトのアップロード」を参照してください。
7. (オプション) 他のウェブサイトコンテンツをバケットにアップロードします。

ステップ 7: エラードキュメントのアップロード

バケットに対して静的ウェブサイトホスティングを有効にするときは、エラードキュメントの名前 (例: **404.html**) を入力します。バケットに対して静的ウェブサイトホスティングを有効にしたら、エラードキュメント名を含む HTML ファイルをバケットにアップロードします。

エラードキュメントを設定するには

1. エラードキュメント (例: 404.html) を作成します。
2. エラードキュメントのファイルをローカルに保存します。

エラードキュメントの名前は、大文字と小文字を区別し、静的ウェブサイトホスティングを有効にする際に入力した名前と厳密に一致している必要があります。たとえば、[Static website hosting] (静的ウェブサイトホスティング) ダイアログボックスの [Error document] (エラードキュメント) 名に `404.html` と入力する場合、エラードキュメントのファイル名も `404.html` である必要があります。

3. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
4. [バケット] リストで、静的ウェブサイトホスティングに使用するバケットの名前を選択します。
5. バケットに対して静的ウェブサイトホスティングを有効にし、エラードキュメントの正確な名前 (例: `404.html`) を入力します。詳細については、ウェブサイトのホスティングの有効化およびカスタムエラードキュメントの設定を参照してください。

   静的ウェブサイトホスティングを有効にしたら、ステップ 6 に進みます。
6. エラードキュメントをバケットにアップロードするには、以下のいずれかを実行します。

   - エラードキュメントファイルをコンソールバケットのリストにドラッグアンドドロップします。
   - [Upload] を選択し、プロンプトに従ってインデックスファイルを選択してアップロードします。

   手順については、「オブジェクトのアップロード」を参照してください。

ステップ 8: S3 のパブリックアクセスのブロック設定を編集する

この例では、ドメインバケット (`example.com`) のパブリックアクセスブロック設定を編集して、パブリックアクセスを許可します。

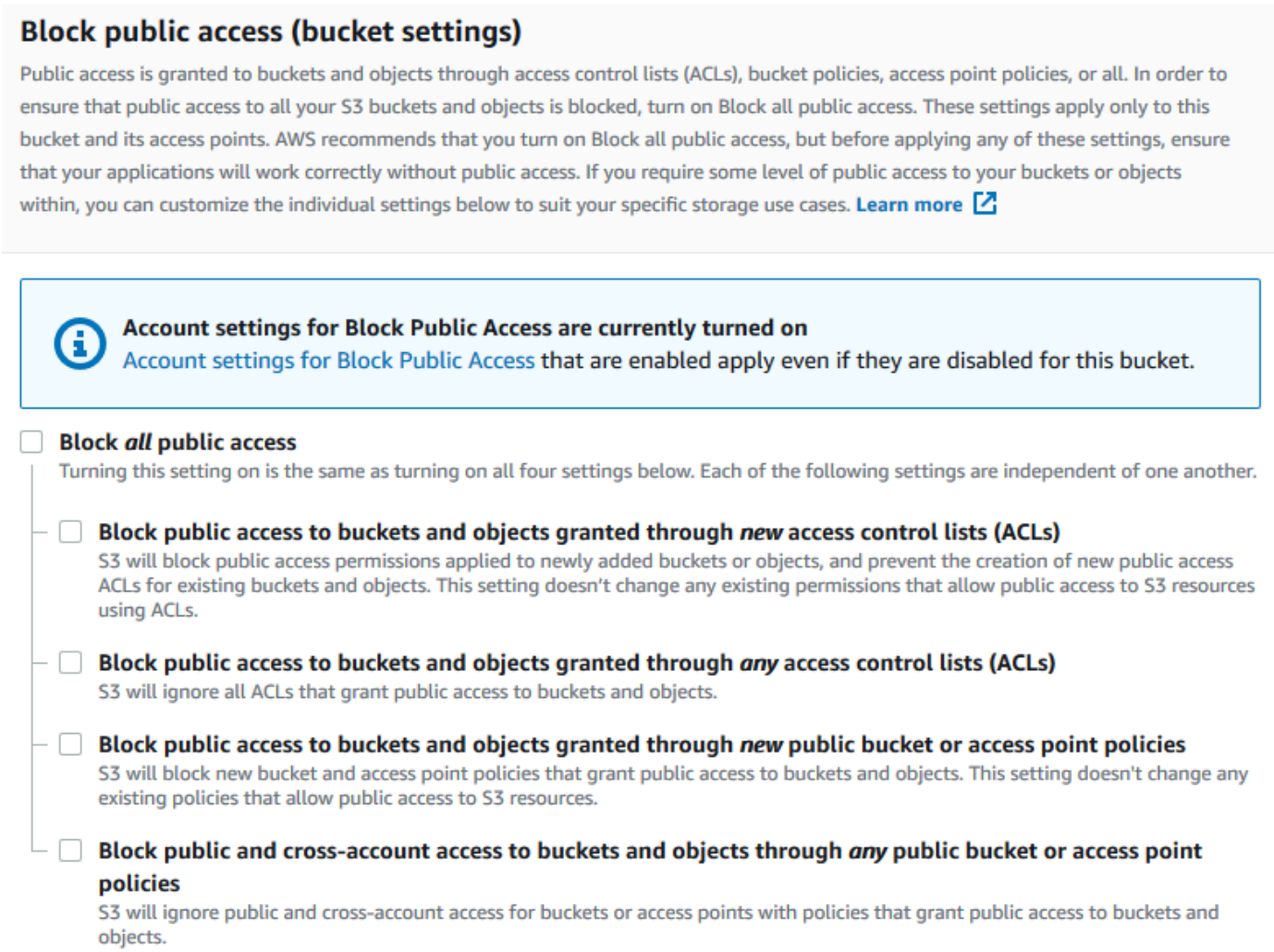
デフォルトでは、Amazon S3 はアカウントとバケットへのパブリックアクセスをブロックします。バケットを使用して静的ウェブサイトをホストする場合は、以下のステップを使用して、パブリックアクセスブロック設定を編集できます。

> **Warning**
>
> このステップを完了する前に「Amazon S3 ストレージへのパブリックアクセスのブロック」を読んで、パブリックアクセスを許可することに伴うリスクを理解し、了承します。パブリックアクセスブロック設定をオフにしてバケットをパブリックにすると、インターネット

上のだれでもバケットにアクセスできるようになります。バケットへのすべてのパブリックアクセスをブロックすることをお勧めします。

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 静的ウェブサイトとして設定されたバケットの名前を選択します。
3. [Permissions (アクセス許可)] を選択します。
4. [ブロックパブリックアクセス (バケット設定)] で [編集] を選択します。
5. [Block all public access (すべてのパブリックアクセスをブロックする)] をクリアし、[Save changes (変更の保存)] を選択します。

> **Warning**
>
> このステップを完了する前に、「Amazon S3 ストレージへのパブリックアクセスのブロック」を確認し、パブリックアクセスの許可に伴うリスクを理解したうえで了承してください。パブリックアクセスブロック設定をオフにしてバケットをパブリックにすると、インターネット上のだれでもバケットにアクセスできるようになります。バケットへのすべてのパブリックアクセスをブロックすることをお勧めします。



Amazon S3 は、バケットのパブリックアクセスブロック設定をオフにします。パブリックで静的ウェブサイトを作成するには、バケットポリシーを追加する前に、アカウントのブロックパブリックアクセス設定を編集する必要があります。パブリックアクセスのブロックのアカウント設定が現在有効になっている場合は、[Block public access (bucket settings) (パブリックアクセスのブロック (バケット設定))] の下にメモが表示されます。

ステップ 9: バケットポリシーをアタッチする

この例では、バケットポリシーをドメインバケット (`example.com`) にアタッチして、パブリック読み取りアクセスを許可します。例えば、バケットポリシーの例の *`Bucket-Name`* は、ドメインバケットの名前 (例: `example.com`) に置き換えます。

S3 のパブリックアクセスブロック設定を編集した後で、バケットへのパブリック読み取りアクセスを許可するバケットポリシーを追加できます。パブリック読み取りアクセスを許可すると、インターネット上のだれでもバケットにアクセスできるようになります。

> **Important**
>
> 次のポリシーは、単なる例として、バケットのコンテンツへのフルアクセスを許可します。このステップに進む前に、「Amazon S3 バケット内のファイルを保護するにはどうすればよいですか?」を確認して、S3 バケット内のファイルを保護するためのベストプラクティスと、パブリックアクセスの許可に伴うリスクを理解してください。

1. [バケット] で、バケットの名前を選択します。
2. [Permissions (アクセス許可)] を選択します。
3. [Bucket Policy (バケットポリシー)] で [編集] を選択します。
4. ウェブサイトのパブリック読み取りアクセスを許可するには、次のバケットポリシーをコピーし、[Bucket policy editor (バケットポリシーエディター)] に貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. `Resource` をバケット名に更新します。

   前述のバケットポリシーの例では、*`Bucket-Name`* はバケット名のプレースホルダーです。このバケットポリシーを独自のバケットで使用するには、バケット名に合わせてこの名前を更新する必要があります。

6. [Save changes] (変更の保存) をクリックします。

   バケットポリシーが正常に追加されたことを示すメッセージが表示されます。

   `Policy has invalid resource` というエラーが表示された場合は、バケットポリシー内のバケット名がバケット名と一致していることを確認します。バケットポリシーの追加については、「S3 バケットポリシーを追加する方法」を参照してください。

   エラーメッセージが表示され、バケットポリシーを保存できない場合は、アカウントとバケットの [パブリックアクセスをブロックする] 設定をチェックして、バケットへのパブリックアクセスを許可していることを確認します。

次のステップでは、ウェブサイトのエンドポイントを判別し、ドメインエンドポイントをテストできます。

ステップ 10: ドメインエンドポイントをテストする

パブリックウェブサイトをホストするようにドメインバケットを設定したら、エンドポイントをテストできます。詳細については、「ウェブサイトエンドポイント」を参照してください。サブドメインバケットは、静的ウェブサイトホスティングではなくウェブサイトリダイレクト用に設定されているため、テストできるのはドメインバケットのエンドポイントのみです。

> **Note**
>
> Amazon S3 は、ウェブサイトへの HTTPS アクセスをサポートしていません。HTTPS を使用する場合は、Amazon CloudFront を使用して Amazon S3 でホストされている静的ウェブサイトを提供できます。
> 詳細については、「CloudFront を使用して、Amazon S3 でホストされた静的ウェブサイトを公開するにはどうすればよいですか?」と「ビューワーと CloudFront との通信で HTTPS を必須にする」を参照してください。

1. [バケット] で、バケットの名前を選択します。
2. [プロパティ] を選択します。
3. ページの下部の [静的ウェブサイトホスティング] で、[Bucket website endpoint (バケットウェブサイトエンドポイント)] を選択します。

   インデックスドキュメントが別のブラウザウィンドウで開きます。

次のステップでは、顧客が両方のカスタム URL を使用してサイトにアクセスできるように、Amazon Route 53 を使用します。

ステップ 11: ドメインとサブドメインのエイリアスレコードを追加する

このステップでは、ドメインマップ `example.com` と `www.example.com` のホストゾーンに追加するエイリアスレコードを作成します。IP アドレスを使用する代わりに、このエイリアスレコードでは Amazon S3 ウェブサイトエンドポイントが使用されます。Amazon Route 53 によって、エイリアスレコードと、Amazon S3 バケットが存在する IP アドレスとのマッピングが維持されます。ルートドメイン用とサブドメイン用の 2 つのエイリアスレコードを作成します。

ルートドメインとサブドメインのエイリアスレコードの追加

ルートドメインのエイリアスレコードを追加するには (**`example.com`**)

1. Route 53 コンソール (https://console.aws.amazon.com/route53/) を開きます。

   > **Note**
   >
   > Route 53 をまだ使用していない場合は、Amazon Route 53 デベロッパーガイドの「ステップ 1: ドメインの登録」を参照してください。セットアップが完了したら、指示を再開できます。

2. [Hosted zones (ホストゾーン)] を選択します。
3. ホストゾーンのリストで、ドメイン名に一致するホストゾーンの名前を選択します。
4. [Create record (レコードを作成)] を選択します。
5. [Switch to wizard] (ウィザードに切り替える) を選択します。

   > **Note**
   >
   > クイック作成を使用してエイリアスレコードを作成する場合は、「Route 53 を設定して S3 バケットにトラフィックをルーティングする」を参照してください。

6. [Simple routing (シンプルルーティング)] を選択して、[Next (次へ)] を選択します。
7. [シンプルなレコードを定義] を選択します。
8. [レコード名] では、デフォルト値をそのまま使用します。これが、ホストゾーンとドメインの名前です。

9. [Value/Route traffic to (値/トラフィックのルーティング先)] で、[Alias to S3 website endpoint (S3 ウェブサイトエンドポイントへのエイリアス)] を選択します。
10. リージョンを選択します。
11. S3 バケットを選択します。

    バケット名は、[Name (名前)] ボックスに表示されている名前と一致する必要があります。[Choose S3 bucket] (S3 バケットを選択) リストに、バケットが作成されたリージョンの Amazon S3 ウェブサイトエンドポイントとともにバケット名が表示されます (例: `s3-website-us-west-1.amazonaws.com (example.com)`)。

    次の場合は、バケットをリストする S3 バケットを選択します。

    - バケットを静的ウェブサイトとして設定した場合。
    - バケットの名前が、作成するレコードの名前と同じである場合。
    - 現在の AWS アカウント でバケットを作成した場合。

    バケットが [Choose S3 bucket] (S3 バケットの選択) リストに表示されない場合は、バケットが作成されたリージョンの Amazon S3 ウェブサイトエンドポイント (例: **`s3-website-us-west-2.amazonaws.com`**) を入力します。Amazon S3 ウェブサイトエンドポイントの完全なリストについては、「Amazon S3 ウェブサイトエンドポイント」を参照してください。エイリアス先の詳細については、Amazon Route 53 デベロッパーガイドの「値/トラフィックのルーティング先」を参照してください。
12. [レコードタイプ] で、[A - IPv4 アドレスと一部の AWS リソースにトラフィックをルーティングします] を選択します。
13. [Evaluate target health (ターゲットの正常性の評価)] で [No (いいえ)] を選択します。
14. [シンプルなレコードを定義] を選択します。

サブドメインのエイリアスレコードを追加するには (**`www.example.com`**)

1. [Configure records] (レコードを設定) で、[Define simple record] (シンプルなレコードを定義) を選択します。
2. サブドメインの [Record name (レコード名)] に「`www`」と入力します。
3. [Value/Route traffic to (値/トラフィックのルーティング先)] で、[Alias to S3 website endpoint (S3 ウェブサイトエンドポイントへのエイリアス)] を選択します。
4. リージョンを選択します。

5. S3 バケットを選択します (例: `s3-website-us-west-2.amazonaws.com (www.example.com)`)。

   バケットが [Choose S3 bucket] (S3 バケットの選択) リストに表示されない場合は、バケットが作成されたリージョンの Amazon S3 ウェブサイトエンドポイント (例: **`s3-website-us-west-2.amazonaws.com`**) を入力します。Amazon S3 ウェブサイトエンドポイントの完全なリストについては、「Amazon S3 ウェブサイトエンドポイント」を参照してください。エイリアス先の詳細については、Amazon Route 53 デベロッパーガイドの「値/トラフィックのルーティング先」を参照してください。
6. [レコードタイプ] で、[A - IPv4 アドレスと一部の AWS リソースにトラフィックをルーティングします] を選択します。
7. [Evaluate target health (ターゲットの正常性の評価)] で [No (いいえ)] を選択します。
8. [シンプルなレコードを定義] を選択します。
9. [Configure records] (レコードを設定) ページで、[Create records] (レコードを作成) を選択します。

> **Note**
>
> 通常、変更は 60 秒以内にすべての Route 53 サーバーに伝播されます。伝播が完了すると、この手順で作成したエイリアスレコードの名前を使用して、トラフィックを Amazon S3 バケットにルーティングできます。

ルートドメインとサブドメインのエイリアスレコードの追加 (以前の Route 53 コンソール)

ルートドメインのエイリアスレコードを追加するには (**`example.com`**)

Route 53 コンソールは再設計されました。Route 53 コンソールでは、暫定的に以前のコンソールを使用することもできます。以前の Route 53 コンソールを使用する場合は、以下の手順に従ってください。

1. Route 53 コンソール (https://console.aws.amazon.com/route53/) を開きます。

> **Note**
>
> Route 53 をまだ使用していない場合は、Amazon Route 53 デベロッパーガイドの「ステップ 1: ドメインの登録」を参照してください。セットアップが完了したら、指示を再開できます。

2. [ホストゾーン] を選択します。
3. ホストゾーンのリストで、ドメイン名に一致するホストゾーンの名前を選択します。
4. [Create Record Set (レコードセットの作成)] を選択します。
5. 次の値を指定します。

   名前

   ホストゾーンとドメインの名前であるデフォルト値を受け入れます。

   ルートドメインの場合、[Name (名前)] フィールドに追加情報を入力する必要はありません。

   タイプ

   [A – IPv4 address (A – IPv4 address)] を選択します。

   エイリアス

   [Yes] を選択します。

   エイリアス先

   リストの [S3 website endpoints (S3 ウェブサイトエンドポイント)] セクションで、バケット名を選択します。

   バケット名は、[Name (名前)] ボックスに表示されている名前と一致する必要があります。[Alias Target (エイリアス先)] リストでは、バケット名の後にはバケットが作成されたリージョンの Amazon S3 ウェブサイトエンドポイントが続きます (例: `example.com (s3-website-us-west-2.amazonaws.com)`)。以下の場合は、[Alias Target] にバケットが表示されます。

   - バケットを静的ウェブサイトとして設定した場合。
   - バケットの名前が、作成するレコードの名前と同じである場合。
   - 現在の AWS アカウント でバケットを作成した場合。

バケットが [エイリアス先] リストに表示されない場合は、バケットが作成されたリージョンの Amazon S3 ウェブサイトエンドポイント (s3-website-us-west-2 など) を入力します。Amazon S3 ウェブサイトエンドポイントの完全なリストについては、「Amazon S3 ウェブサイトエンドポイント」を参照してください。エイリアス先の詳細については、Amazon Route 53 デベロッパーガイドの「値/トラフィックのルーティング先」を参照してください。

ルーティングポリシー

デフォルト値の [Simple] をそのまま使用します。

ターゲットの正常性の評価

デフォルト値の [No] をそのまま使用します。

6. [Create] (作成) を選択します。

サブドメインのエイリアスレコードを追加するには (**www.example.com**)

1. ルートドメイン (example.com) のホストゾーンで、[Create Record Set (レコードセットの作成)] を選択します。
2. 次の値を指定します。

名前

サブドメインの場合、ボックスに www と入力します。

タイプ

[A – IPv4 address (A – IPv4 address)] を選択します。

エイリアス

[Yes] を選択します。

エイリアス先

リストの [S3 website endpoints (S3 ウェブサイトエンドポイント)] セクションで、[Name (名前)] フィールドに表示される名前と同じバケット名を選択します (例: www.example.com (s3-website-us-west-2.amazonaws.com))。

ルーティングポリシー

デフォルト値の [Simple] をそのまま使用します。

ターゲットの正常性の評価

デフォルト値の [No] をそのまま使用します。

3. [Create] (作成) を選択します。

> **Note**
>
> 通常、変更は 60 秒以内にすべての Route 53 サーバーに伝播されます。伝播が完了すると、この手順で作成したエイリアスレコードの名前を使用して、トラフィックを Amazon S3 バケットにルーティングできます。

ステップ 12: ウェブサイトをテストする

ウェブサイトとリダイレクトが正しく機能することを確認します。ブラウザで、URL を入力します。この例では、次の URL を試すことができます。

- ドメイン (`http://example.com`) – `example.com` バケット内のインデックスドキュメントを表示します。
- サブドメイン (`http://www.example.com`) – リクエストを `http://example.com` にリダイレクトします。`example.com` バケット内のインデックスドキュメントが表示されます。

ウェブサイトやリダイレクトのリンクが機能しない場合は、以下のことを試してください。

- キャッシュのクリア– ウェブブラウザのキャッシュをクリアします。
- ネームサーバーの確認– キャッシュをクリアしてもウェブページやリダイレクトのリンクが機能しない場合は、ドメインのネームサーバーとホストゾーンのネームサーバーを比較します。ネームサーバーが一致しない場合は、ドメインのネームサーバーを更新してホストゾーンのネームサーバーと一致させる必要があります。詳細については、「ドメインのネームサーバーおよびグルーレコードの追加あるいは変更」を参照してください。

ルートドメインとサブドメインのテストが完了したら、Amazon CloudFront ディストリビューションを設定できます。これにより、ウェブサイトのパフォーマンスを向上させ、ウェブサイトのトラフィックの確認に使用できるログを指定できます。詳細については、「Amazon CloudFront によるウェブサイトの高速化」を参照してください。

Amazon CloudFront によるウェブサイトの高速化

Amazon CloudFront を使用すると、Amazon S3 ウェブサイトのパフォーマンスを高めることができます。CloudFront を使用すると、世界中のデータセンター (エッジロケーション と呼ばれる) からお客様のウェブサイトのファイル (HTML、画像、動画など) を使用できるようになります。閲覧者がウェブサイトからファイルをリクエストすると、CloudFront は自動的に最も近いエッジロケーションのファイルのコピーにリクエストをリダイレクトします。これにより、遠い場所にあるデータセンターからコンテンツをリクエストした場合よりもダウンロード時間が高速になります。

CloudFront は、指定した期間にわたってコンテンツをエッジロケーションにキャッシュします。閲覧者が有効期限より長くキャッシュされていたコンテンツをリクエストすると、CloudFront はオリジンサーバーをチェックしてより新しいバージョンのコンテンツがあるかどうかを確認します。新しいバージョンが利用可能である場合、CloudFront は新しいバージョンをエッジロケーションにコピーします。元のコンテンツへの変更は、閲覧者がコンテンツをリクエストすると、エッジロケーションにレプリケートされます。

Route 53 なしで CloudFront を使用する

このページのチュートリアルでは Route 53 を使用して CloudFront 配信をポイントしています。ただし、Route 53 を使用せずに CloudFront を使用して Amazon S3 バケットでホストされているコンテンツを配信する場合は、「Amazon CloudFront チュートリアル: Amazon S3 用の動的コンテンツ配信のセットアップ」を参照してください。CloudFront を使用して Amazon S3 バケットでホストされているコンテンツを配信する場合は、任意のバケット名を使用できます。また、HTTP と HTTPS の両方がサポートされます。

AWS CloudFormation テンプレートを使用した設定の自動化

ウェブサイトへの配信を行う CloudFront ディストリビューションを作成する安全な静的ウェブサイトを、AWS CloudFormation テンプレートを使用して設定する方法の詳細については、Amazon CloudFront デベロッパーガイドの「安全な静的ウェブサイトの開始方法」を参照してください。

トピック

ステップ 1: CloudFront ディストリビューションを作成する

まず、CloudFront ディストリビューションを作成します。これにより、ウェブサイトは世界中のデータセンターから利用できるようになります。

Amazon S3 オリジンを使用してディストリビューションを作成するには

1. で CloudFront コンソールを開きますhttps://console.aws.amazon.com/cloudfront/v4/home
2. [Create Distribution] を選択します。
3. [ディストリビューションの作成] ページで、[オリジンの設定] セクションの [オリジンドメイン名] に、バケットの Amazon S3 ウェブサイトエンドポイントを入力します (例: **`example.com.s3-website.us-west-1.amazonaws.com`**)。

   CloudFront により自動的に [Origin ID (オリジン ID)] が入力されます。
4. [Default Cache Behavior Settings (キャッシュ動作のデフォルト設定)] の値はデフォルトのままにしておきます。

   [ビューワープロトコルポリシー] のデフォルト設定では、静的ウェブサイトに HTTPS を使用できます。これらの設定オプションの詳細については、Amazon CloudFront 開発者ガイドの「ウェブディストリビューションを作成または更新する場合に指定する値」を参照してください。
5. [Distribution Settings (ディストリビューションの設定)] で、以下の作業を行います。

   a. [Price Class (価格クラス)] は [Use All Edge Locations (Best Performance) (すべてのエッジロケーションを使用する (最適なパフォーマンス))] に設定されたままにしておきます。

   b. [代替ドメイン名 (CNAME)] をルートドメインと `www` サブドメインに設定します。このチュートリアルでは、`example.com` と `www.example.com` です。

   > **Important**
   >
   > このステップを実行する前に、代替ドメイン名を使用するための要件、特に有効な SSL/TLS 証明書の必要性について確認します。

   c. [SSL 証明書] で、[Custom SSL Certificate (example.com) (独自 SSL 証明書 (example.com))] を選択し、ドメイン名とサブドメイン名を対象とするカスタム証明書を選択します。

詳細については、Amazon CloudFront 開発者ガイドの「SSL 証明書」を参照してください。

d. [Default Root Object (デフォルトのルートオブジェクト)] に、インデックスドキュメントの名前を入力します (例: `index.html`)。

ディストリビューションにアクセスするために使用される URL にファイル名が含まれていない場合、CloudFront ディストリビューションはインデックスドキュメントを返します。[Default Root Object (デフォルトのルートオブジェクト)] は、静的ウェブサイトのインデックスドキュメントの名前と正確に一致する必要があります。詳細については、「インデックスドキュメントの設定」を参照してください。

e. [ログ記録] を [On (オン)] に設定します。

> **Important**
>
> ディストリビューションを作成または更新して CloudFront のログを有効にすると、CloudFront はバケットのアクセスコントロールリスト (ACL) を更新して、バケットにログを書き込む `awslogsdelivery` アクセス許可を `FULL_CONTROL` アカウントに付与します。詳細については、「Amazon CloudFront デベロッパーガイド」の「標準ログ記録の設定とログファイルへのアクセスに必要なアクセス許可」を参照してください。ログを保存するバケットが S3 オブジェクト所有権のバケット所有者の強制設定を使用して ACL を無効にすると、CloudFront はバケットにログを書き込むことができません。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

f. [Bucket for Logs (ログ用のバケット)] で、作成したログ記録バケットを選択します。

ログ記録バケットの設定の詳細については、「(オプション) ウェブトラフィックのログ記録」を参照してください。

g. CloudFront ディストリビューションへのトラフィックによって生成されたログをフォルダに保存する場合は、[Log Prefix (ログのプレフィックス)] にフォルダ名を入力します。

h. 他の設定はすべてデフォルト値のままにしておきます。

6. [Create Distribution] を選択します。

7. ディストリビューションのステータスを表示するには、コンソールでディストリビューションを見つけて、[ステータス] 列を確認します。

[InProgress] のステータスは、ディストリビューションがまだ完全にデプロイされていないことを示します。

ディストリビューションがデプロイされたら、新しい CloudFront ドメイン名でコンテンツを参照できます。

8. CloudFront コンソールに表示される [Domain Name (ドメイン名)] の値を書き留めます (例: `dj4p1rv6mvubz.cloudfront.net`)。
9. CloudFront ディストリビューションが正常に動作していることを確認するには、ウェブブラウザでディストリビューションのドメイン名を入力します。

   ウェブサイトが表示されている場合、CloudFront ディストリビューションは機能します。ウェブサイトに Amazon Route 53 で登録されたカスタムドメインがある場合は、次のステップでレコードセットを更新するために CloudFront ドメイン名が必要になります。

ステップ 2: ドメインとサブドメインのレコードセットを更新する

CloudFront ディストリビューションが正常に作成されたので、Route 53 でエイリアスレコードを更新して、新しい CloudFront ディストリビューションを指すようにします。

CloudFront ディストリビューションを指すようにエイリアスレコードを更新するには

1. Route 53 コンソール (https://console.aws.amazon.com/route53/) を開きます。
2. 左のナビゲーションで、[Hosted zones] (ホストゾーン) を選択します。
3. [Hosted Zones] (ホストゾーン) ページで、サブドメイン用に作成したホストゾーンを選択します (例: `www.example.com`)。
4. [Records] (レコード) で、サブドメイン用に作成した A レコードを選択します。
5. [Record details] (レコードの詳細) で、[Edit record] (レコードを編集) を選択します。
6. [Route traffic to] (トラフィックのルーティング先) で、[Alias to CloudFront distribution] (CloudFront ディストリビューションへのエイリアス) を選択します。
7. [Choose distribution] (ディストリビューションを選択) で、CloudFront ディストリビューションを選択します。
8. [Save] を選択します。
9. ルートドメインの A レコードを CloudFront ディストリビューションにリダイレクトするには、ルートドメイン用のこの手順を繰り返します (例: `example.com`)。

レコードセットの更新は、2～48 時間以内に有効になります。

10. 新しい A レコードが有効かどうかを確認するには、ウェブブラウザで、サブドメイン URL を入力します (例: `http://www.example.com`)。

    ブラウザがルートドメイン (例: `http://example.com`) にリダイレクトしなくなった場合、新しい A レコードが有効になっています。新しい A レコードが有効になると、新しい A レコードによって CloudFront ディストリビューションにルーティングされるトラフィックが、ルートドメインにリダイレクトされなくなります。`http://example.com` または`http://www.example.com` を使用してサイトを参照する閲覧者は、最も近い CloudFront エッジロケーションにリダイレクトされるため、ダウンロード時間が短縮されます。

> **Tip**
>
> ブラウザはリダイレクト設定をキャッシュできます。新しい A レコードの設定が有効になったと考えられるにもかかわらず、ブラウザによって`http://www.example.com` がまだ `http://example.com` にリダイレクトされる場合は、ブラウザアプリケーションを閉じてから再び開いてブラウザの履歴とキャッシュをクリアするか、異なるウェブブラウザを使用してみてください。

(オプション) ステップ 3: ログファイルを確認する

アクセスログにより、ウェブサイトを訪れたユーザーの数がわかります。また、Amazon EMR など他のサービスで分析できる貴重なビジネスデータも含まれています。

CloudFront ログは、CloudFront ディストリビューションを作成してログを有効にするときに選択したバケットとフォルダに保存されます。CloudFront は、対応するリクエストが行われると、24 時間以内にログバケットにログを書き込みます。

ウェブサイトのログファイルを表示するには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. ウェブサイトのログ記録バケットを選択します。
3. CloudFront ログフォルダを選択します。
4. CloudFront によって書き込まれた `.gzip` ファイルを開く前に、まずファイルをダウンロードします。

学習のためだけにウェブサイトを作成した場合は、割り当てたリソースを削除して、課金を停止できます。これを行うには、「サンプルリソースのクリーンアップ」を参照してください。AWS リソースを削除すると、ウェブサイトは使用できなくなります。

サンプルリソースのクリーンアップ

学習のために静的ウェブサイトを作成した場合は、割り当てた AWS リソースをここで削除して、課金を停止します。AWS リソースを削除すると、ウェブサイトは使用できなくなります。

タスク

- ステップ 1: Amazon CloudFront ディストリビューションを削除する
- ステップ 2: Route 53 ホストゾーンを削除する
- ステップ 3: ログ記録を無効にし、S3 バケットを削除する

ステップ 1: Amazon CloudFront ディストリビューションを削除する

Amazon CloudFront ディストリビューションを削除する前に、そのディストリビューションを無効にする必要があります。無効になったディストリビューションは機能しなくなり、料金も発生しません。無効にしたディストリビューションはいつでも有効にすることができます。無効にしたディストリビューションを削除すると、使用できなくなります。

CloudFront ディストリビューションを無効にして削除するには

1. で CloudFront コンソールを開きますhttps://console.aws.amazon.com/cloudfront/v4/home
2. 無効にするディストリビューションを選択してから [Disable (無効化)] を選択します。
3. 確認を求められたら、[Yes, Disable (はい、無効化する)] を選択します。
4. 無効にしたディストリビューションを選択してから [削除] を選択します。
5. 確認を求めるメッセージが表示されたら、[Yes, Delete (はい、削除します)] を選択します。

ステップ 2: Route 53 ホストゾーンを削除する

ホストゾーンを削除する前に、作成したレコードセットを削除する必要があります。NS レコードと SOA レコードを削除する必要はありません。これらはホストゾーンが削除されるときに自動的に削除されます。

レコードセットを削除するには

1. Route 53 コンソール (https://console.aws.amazon.com/route53/) を開きます。
2. ドメイン名のリストでドメイン名を選択し、[Go to Record Sets (レコードセットに移動)] を選択します。
3. レコードセットのリストで、作成した A レコードを選択します。

   各レコードセットのタイプは [タイプ] 列に表示されます。
4. [Delete Record Set (レコードセットの削除)] を選択します。
5. 確認を求められたら、[確認] を選択します。

Route 53 ホストゾーンを削除するには

1. 前のステップを終了した画面で [Back to Hosted Zones (ホストゾーンに戻る)] を選択します。
2. ドメイン名を選択してから [Delete Hosted Zone (ホストゾーンの削除)] を選択します。
3. 確認を求められたら、[確認] を選択します。

ステップ 3: ログ記録を無効にし、S3 バケットを削除する

S3 バケットを削除する前に、バケットのログ記録が無効であることを確認します。それ以外の場合、削除するバケットへのログの書き込みが AWS によって継続されます。

バケットのログ記録を無効にするには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. [バケット] でバケット名を選択し、[プロパティ] を選択します。
3. [プロパティ] から [ログ記録] を選択します。
4. [有効] チェックボックスをオフにします。
5. [Save] を選択します。

これで、バケットを削除できます。詳細については、「バケットの削除」を参照してください。

Amazon S3 バケットの作成、設定、操作

Amazon S3 にデータを保存するには、バケットおよびオブジェクトと呼ばれるリソースを使用します。バケット とは、オブジェクトのコンテナのことです。オブジェクト とは、ファイルと、そのファイルを記述している任意のメタデータのことです。

Amazon S3 にオブジェクトを保存するには、バケットを作成してから、オブジェクトをバケットにアップロードします。オブジェクトがバケットの中にあるときは、オブジェクトを開き、ダウンロードして、移動させます。オブジェクトまたはバケットが不要になったら、リソースをクリーンアップします。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

> **Note**
>
> Amazon S3 では、お支払いは実際に使用した分のみです。Amazon S3 の機能と料金の詳細については、「Amazon S3」を参照してください。Amazon S3 の新規のお客様は、Amazon S3 を無料で使い始めることができます。詳細については、[AWS 無料利用枠] を参照してください。

このセクションのトピックでは、Amazon S3 でのバケット使用の概要について説明します。バケットの命名、作成、アクセス、削除に関する情報が含まれます。バケット内でのオブジェクトの表示および一覧表示について詳しくは、オブジェクトの整理、リスト化、使用 を参照してください。

トピック

- バケットを空にする
- バケットの削除
- Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定
- Mountpoint for Amazon S3 の使用
- Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定
- ストレージ転送と使用量のリクエスタ支払いバケットの使用
- バケットの制約と制限

バケットの概要

Amazon S3 にデータ (写真、動画、ドキュメントなど) をアップロードするには、いずれかの AWS リージョン に S3 バケットを作成する必要があります。

バケットとは、Amazon S3 に保存されるオブジェクトのコンテナです。バケットにはオブジェクトをいくつでも保存でき、アカウントにはバケットを 100 個まで保存できます。増加をリクエストするには、Service Quotas コンソールにアクセスしてください。

すべてのオブジェクトはバケット内に保存されます。例えば、photos/puppy.jpg という名前のオブジェクトが米国西部 (オレゴン) リージョンにある DOC-EXAMPLE-BUCKET バケットに保存される場合、URL https://DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com/photos/puppy.jpg を使用してアドレスを解決できます。詳細については、「バケットへのアクセス」を参照してください。

実装の面では、バケットとオブジェクトは AWS リソースであり、Amazon S3 はバケットやオブジェクトを管理するための API を提供します。たとえば、Amazon S3 API を使用してバケットの作成やオブジェクトのアップロードを行うことができます。これらの操作は Amazon S3 コンソールを使用して実行することもできます。コンソールは、Amazon S3 API を使用して Amazon S3 にリクエストを送信します。

このセクションでは、バケットを操作する方法について説明します。オブジェクトの操作方法の詳細については、「Amazon S3 オブジェクトの概要」を参照してください。

Amazon S3 はグローバルバケットをサポートしています。つまり、各バケット名が、パーティション内のすべての AWS リージョン のすべての AWS アカウント で一意である必要があることを意味します。パーティションは、リージョンのグループです。AWS には、現在、aws (標準リージョン)、aws-cn (中国リージョン)、および aws-us-gov (AWS GovCloud (US)) の 3 つのパーティションがあります。

バケットが作成された後は、バケットが削除されるまで、同じパーティションの別の AWS アカウント でそのバケットの名前を使用することはできません。可用性やセキュリティ検証の目的で、特定のバケット命名規則に依存しないでください。バケットの命名のガイドラインについては、バケットの名前付け を参照してください。

Amazon S3 は、指定したリージョンでバケットを作成します。レイテンシーを減らしてコストを最小化し、規制要件に対応するには、地理的に近い AWS リージョン を選択します。たとえば、ヨーロッパにお住まいの場合は、欧州 (アイルランド) または欧州 (フランクフルト) リージョンにバケットを作成するとよいでしょう。Amazon S3 のリージョンの一覧については、AWS 全般のリファレンスのリージョンとエンドポイントを参照してください。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

> **Note**
>
> 特定の AWS リージョン に作成したバケットに属するオブジェクトは、お客様が明示的に他のリージョンに転送しない限り、そのリージョンから移動しません。たとえば、欧州 (アイルランド) リージョンに保存されているオブジェクトは、ずっとそのリージョンに置かれたままです。

トピック

- アクセス許可について
- バケットへのパブリックアクセスを管理する
- バケット設定オプション

アクセス許可について

AWS アカウントのルートユーザーの認証情報を使用して、バケットを作成したり、その他の Amazon S3 オペレーションを実行したりできます。ただし、バケット作成などのリクエストを行う目的で AWS アカウント のルートユーザーの認証情報を使用しないことをお勧めします。代わり

に、AWS Identity and Access Management (IAM) ユーザーを作成し、そのユーザーにフルアクセスを許可します (デフォルトではユーザーにアクセス許可はありません)。

これらのユーザーは、管理者と呼ばれます。アカウントのルートユーザーの認証情報ではなく、管理者ユーザーの認証情報を使用して、AWS を操作し、バケットの作成、ユーザーの作成、および許可の付与などのタスクを実行できます。

詳細については、AWS 全般のリファレンスの「AWS アカウントのルートユーザー認証情報と IAM ユーザー認証情報」と IAM ユーザーガイドの「IAM でのセキュリティのベストプラクティス」を参照してください。

リソースを作成する AWS アカウント は、そのリソースを所有しています。例えば、AWS アカウント に IAM ユーザーを作成し、そのユーザーにバケットを作成する許可を付与すると、ユーザーはバケットを作成できます。しかし、ユーザーはバケットを所有していません。ユーザーが属する AWS アカウント がバケットを所有しています。ユーザーがその他のバケットオペレーションを実行するには、リソース所有者から追加のアクセス許可を取得する必要があります。Amazon S3 リソースのアクセス許可を管理する方法の詳細については、「Amazon S3 用 Identity and Access Management」を参照してください。

バケットへのパブリックアクセスを管理する

パブリックアクセスは、アクセスコントロールリスト (ACL)、バケットポリシー、またはその両方からバケットおよびオブジェクトに付与されます。Amazon S3 には、パブリックアクセスをブロックする設定があり、Amazon S3 リソースへのパブリックアクセスを管理するのに役立ちます。Amazon S3 Block Public Access 設定では ACL およびバケットポリシーを上書きできるため、これらのリソースへのパブリックアクセスに均一な制限を適用できます。Block Public Access 設定は、個々のバケットまたはアカウント内のすべてのバケットに適用できます。

すべての Amazon S3 バケットとオブジェクトのパブリックアクセスを確実にブロックするように、バケットを作成するときは、デフォルトで 4 つすべての [パブリックアクセスをブロック] 設定が有効になっています。アカウントに対して [パブリックアクセスをブロック] の 4 つの設定をすべてオンにすることをお勧めします。これらの設定によって、現在および将来のバケットのパブリックアクセスはすべてブロックされます。

これらの設定を適用する前に、アプリケーションがパブリックアクセスなしで正しく動作することを確認してください。「Amazon S3 を使用して静的ウェブサイトをホスティングする」に示す静的なウェブサイトをホストする場合など、バケットやオブジェクトにある程度のパブリックアクセスが必要な場合は、ストレージのユースケースに合わせて個別に設定をカスタマイズできます。詳細については、「Amazon S3 ストレージへのパブリックアクセスのブロック」を参照してください。

ただし、[パブリックアクセスをブロック] 設定は、有効にしておくことを強くお勧めします。4 つすべての [パブリックアクセスをブロック] 設定を有効にしたまま、静的ウェブサイトをホストする場合は、Amazon CloudFront のオリジンアクセスコントロール (OAC) を使用できます。Amazon CloudFront は、セキュアな静的ウェブサイトをセットアップするために必要な機能を提供します。Amazon S3 静的ウェブサイトは、HTTP エンドポイントのみをサポートしています。Amazon CloudFront は、耐久性に優れた Amazon S3 のあるストレージを使用し、HTTPS などの、追加のセキュリティヘッダーを提供します。HTTPS では、通常の HTTP リクエストを暗号化し、一般的なサイバー攻撃から保護することで、セキュリティが強化されます。

詳細については、「Amazon CloudFront デベロッパーガイド」の「安全な静的ウェブサイトの使用開始」を参照してください。

> **Note**
>
> バケットとそのパブリックアクセス設定を一覧表示したときに `Error` が表示された場合は、必要なアクセス許可がない可能性があります。以下のアクセス許可がユーザーポリシーまたはロールポリシーに追加されていることを確認します。
>
> ```
> s3:GetAccountPublicAccessBlock
> s3:GetBucketPublicAccessBlock
> s3:GetBucketPolicyStatus
> s3:GetBucketLocation
> s3:GetBucketAcl
> s3:ListAccessPoints
> s3:ListAllMyBuckets
> ```
>
> まれに、AWS リージョンの停止が原因で、リクエストが失敗することもあります。

バケット設定オプション

Amazon S3 では、バケットを設定するためのさまざまなオプションをサポートしています。たとえば、バケットをウェブサイトのホスティング用に設定する、バケット内のオブジェクトのライフサイクルを管理するための設定を追加する、およびバケットへのアクセスをすべて記録するようにバケットを設定することができます。Amazon S3 では、バケットの設定情報を保存および管理するためのサブリソースをサポートしています。Amazon S3 API を使用して、これらのサブリソースを作成および管理できます。ただし、コンソールや AWS SDK を使用することもできます。

> **Note**
>
> また、オブジェクトレベルの設定もあります。たとえば、オブジェクトに固有のアクセスコントロールリスト (ACL) を設定することによって、オブジェクトレベルのアクセス許可を設定できます。

これらは、特定のバケットやオブジェクトのコンテキストに存在するため、サブリソースと呼ばれます。次の表に、バケット固有の設定を管理するためのサブリソースのリストを示します。

サブリソース	説明
cors (Cross-Origin Resource Sharing)	クロスオリジンリクエストを許可するようにバケットを設定できます。 詳細については、「Cross-Origin Resource Sharing (CORS) の使用」を参照してください。
イベントの通知	指定したバケットイベントの通知を送信するようにバケットを設定できます。 詳細については、「Amazon S3 イベント通知」を参照してください。
ライフサイクル	バケットのライフサイクルが明確に定義されていれば、バケット内のオブジェクトのライフサイクルルールを定義できます。たとえば、作成されてから 1 年後にオブジェクトアーカイブする、作成されてから 10 年後にオブジェクトを削除するなどのルールを定義できます。 詳細については、「ストレージのライフサイクルの管理」を参照してください。
location	バケットを作成するときには、Amazon S3 でバケットを作成する AWS リージョン を指定します。Amazon S3 では、この情報を location サブリソースに格納し、この情報を取得するための API をお客様に提供します。
logging	ログ記録によって、バケットへのアクセスのリクエストを追跡できます。各アクセスログレコードには、1 つのアクセスリクエストに関する詳細が含まれます。内容は、リクエスタ、バケット名、リクエスト時刻、リクエストアクション、応答ステータス、およびエラーコード (存在する場合) です。アク

サブリソース	説明
	セスログ情報は、セキュリティとアクセス監査に役立ちます。また、顧客基盤について知り、Amazon S3 の請求を理解することにも役立ちます。 詳細については、「サーバーアクセスログによるリクエストのログ記録」を参照してください。
オブジェクトのロック	S3 のオブジェクトのロックを使用するには、バケットに対して有効にする必要があります。オプションで、バケットに配置された新しいオブジェクトに適用されるデフォルトのリテンションモードと期間を設定することもできます。 詳細については、「S3 オブジェクトロックの使用」を参照してください。
policy および ACL (アクセスコントロールリスト)	すべてのリソース (バケットやオブジェクトなど) はデフォルトではプライベートです。Amazon S3 では、バケットレベルのアクセス許可を付与および管理するために、バケットポリシーおよびアクセスコントロールリスト (ACL)の両方のオプションをサポートしています。Amazon S3 では、policy および acl サブリソースにアクセス許可情報を格納します。 詳細については、「Amazon S3 用 Identity and Access Management」を参照してください。
レプリケーション	レプリケーションは、同一または異なる AWS リージョン にあるバケット間でオブジェクトを自動的に非同期コピーする機能です。詳細については、「オブジェクトのレプリケーション」を参照してください。
requestPayment	デフォルトでは、バケットを作成した AWS アカウント (バケット所有者) に、バケットからのダウンロードの料金をお支払いいただきます。バケット所有者は、このサブリソースを使用して、ダウンロードをリクエストするユーザーにダウンロードの料金が課金されるように指定できます。Amazon S3 では、このサブリソースを管理するための API も利用できます。 詳細については、「ストレージ転送と使用量のリクエスタ支払いバケットの使用」を参照してください。

サブリソース	説明
タグ付け	バケットにコスト配分タグを追加して、AWS コストを分類して追跡できます。Amazon S3 では、バケットのタグを保存、管理するために、tagging サブリソースを提供しています。タグを使用してバケットに適用すると、AWS によって、使用量とコストがタグごとに集計されたコスト配分レポートが生成されます。 詳細については、「Amazon S3 の請求および使用状況レポート」を参照してください。
Transfer Acceleration	Transfer Acceleration を使用すると、クライアントと S3 バケットの間で、長距離にわたるファイル転送を高速、簡単、安全に行えるようになります。Transfer Acceleration は、Amazon CloudFront の世界中に点在するエッジロケーションを利用します。 詳細については、「Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定」を参照してください。
バージョニング	バージョニングは、誤って上書き、削除した場合の回復に役立ちます。 誤って削除または上書きされたオブジェクトを回復するためのベストプラクティスとしてバージョニングをお勧めします。 詳細については、「S3 バケットでのバージョニングの使用」を参照してください。
website	静的ウェブサイトホスティング用にバケットを設定できます。Amazon S3 では、website サブリソースを作成することによって、この設定を格納します。 詳細については、「Amazon S3 を使用して静的ウェブサイトをホスティングする」を参照してください。

バケットの名前付け

Amazon S3 の汎用バケットとディレクトリバケットの命名については、次の規則が適用されます。

トピック

汎用バケットの命名規則

汎用バケットの命名については、次の命名規則が適用されます。

- バケット名は 3 (最少) ~ 63 (最大) 文字の長さにする必要があります。
- バケット名は、小文字、数字、ドット (.)、およびハイフン (-) のみで構成できます。
- バケット名は、文字または数字で開始および終了する必要があります。
- バケット名には、連続する 2 つのピリオドを含めることはできません。
- バケット名は IP アドレスの形式 (192.168.5.4 など) にはできません。
- バケット名のプレフィックスは `xn--` で始まってはいけません。
- バケット名を、プレフィックス `sthree-` または `sthree-configurator` で始めることはできません。
- バケット名のサフィックスは `-s3alias` で終わってはいけません。このサフィックスは、アクセスポイントのエイリアス名用に予約されています。詳細については、「S3 バケットアクセスポイントでのバケット形式のエイリアスの使用」を参照してください。
- バケット名のサフィックスは `--ol-s3` で終わってはいけません。このサフィックスは、Object Lambda アクセスポイントのエイリアス名用に予約されています。詳細については、「S3 バケット Object Lambda アクセスポイントでのバケット形式のエイリアスの使用方法」を参照してください。
- バケット名は、パーティション内のすべての AWS リージョン のすべての AWS アカウント にわたって一意である必要があります。パーティションは、リージョンのグループです。AWS には、現在、`aws` (標準リージョン)、`aws-cn` (中国リージョン)、および `aws-us-gov` (AWS GovCloud (US)) の 3 つのパーティションがあります。
- バケットが削除されるまで、バケット名を同じパーティション内の別の AWS アカウント で使用することはできません。
- Amazon S3 Transfer Acceleration で使用されるバケットの名前にドット (.) を付けることはできません。Transfer Acceleration の詳細については、「Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定」を参照してください。

互換性を最も高くするには、静的ウェブサイトホスティング専用のバケットを除き、バケット名にドット (.) を使用しないことをお勧めします。バケット名にドットを含めると、証明書の検証を独自

に実行しない限り、HTTPS 経由の仮想ホスト形式のアドレス指定は使用できません。これは、バケット名にドットが含まれていると、バケットの仮想ホスティング用のセキュリティ証明書は機能しないためです。

静的ウェブサイトホスティングは HTTP 経由でのみ使用されるため、この制限は静的ウェブサイトホスティング用のバケットには影響しません。仮想ホスティング形式のアドレス指定の詳細については、「バケットの仮想ホスティング」を参照してください。静的ウェブサイトホスティングの詳細については、「Amazon S3 を使用して静的ウェブサイトをホスティングする」を参照してください。

> **Note**
>
> 2018 年 3 月 1 日までは、米国東部 (バージニア北部) リージョンで作成したバケットの名前として最大 255 文字を使用し、大文字とアンダースコアを含めることができました。2018 年 3 月 1 日以降、米国東部 (バージニア北部) の新しいバケットは、他のすべてのリージョンに適用されている同じルールに準拠する必要があります。

オブジェクトキー名の詳細については、「オブジェクトキー名の作成」を参照してください。

汎用バケット名の例

次の例に示すバケット名は有効であり、汎用バケットの命名についての推奨ガイドラインに従っています。

- `docexamplebucket1`
- `log-delivery-march-2020`
- `my-hosted-content`

次の例に示すバケット名は有効ですが、静的ウェブサイトホスティング以外の用途には推奨されません。

- `docexamplewebsite.com`
- `www.docexamplewebsite.com`
- `my.example.s3.bucket`

次の例に示すバケット名は無効です。

- `doc_example_bucket` (アンダースコアが含まれている)

- `DocExampleBucket` (大文字が含まれている)
- `doc-example-bucket-` (ハイフンで終わっている)

ディレクトリバケットの命名規則

ディレクトリバケット名は、次の条件を満たす必要があります。

- 選択した AWS リージョン とアベイラビリティーゾーン内では一意であること。
- 名前は、サフィックスを含め 3 文字 (最短) から 63 文字 (最長) の長さでなければなりません。
- 小文字の英文字、数字、およびハイフン (-) で構成されていること。
- 文字や数字で始まり、文字や数字で終わります。
- `--azid--x-s3` のサフィックスを含める必要があります。

> **Note**
>
> コンソールを使用してディレクトリバケットを作成すると、指定したベース名にサフィックスが自動的に追加されます。このサフィックスには、選択したアベイラビリティーゾーンのアベイラビリティーゾーン ID が含まれます。
> API を使用してディレクトリバケットを作成する場合は、リクエストでアベイラビリティーゾーン ID を含む完全なサフィックスを指定する必要があります。アベイラビリティーゾーン ID のリストについては、「S3 Express One Zone のアベイラビリティーゾーンとリージョン」を参照してください。

Amazon S3 バケットに対するアクセスと一覧表示

Amazon S3 バケットを一覧表示してアクセスするには、さまざまなツールを使用できます。以下のツールを確認して、どのアプローチが自分のユースケースに合っているかを判断してください。

- Amazon S3 コンソール: Amazon S3 コンソールを使用すると、バケットに簡単にアクセスしてバケットのプロパティを変更できます。コンソール UI を使用すると、コードを記述することなく、ほとんどのバケットオペレーションを実行することもできます。
- AWS CLI: 複数のバケットにアクセスする必要がある場合は、AWS Command Line Interface (AWS CLI) を使用して、一般的で反復的なタスクを自動化することで時間を節約できます。組織

の規模が拡大するにつれて、一般的なアクションのスクリプト作成性と再現性が頻繁に検討されます。詳細については、「AWS CLI を使用した Amazon S3 での開発」を参照してください。

- Amazon S3 REST API: Amazon S3 REST API を使用して独自のプログラムを作成したり、プログラムでバケットにアクセスしたりできます。Amazon S3 は、API アーキテクチャをサポートします。このアーキテクチャでは、バケットやオブジェクトはリソースであり、それぞれリソースを一意に識別するリソース URI を持ちます。詳細については、「REST API を使用した Amazon S3 での開発」を参照してください。

Amazon S3 バケットのユースケースに応じて、バケット内の基になるデータにアクセスするための推奨方法が異なります。以下のリストには、データにアクセスする一般的なユースケースが含まれています。

- 静的ウェブサイト – Amazon S3 を使用して静的ウェブサイトをホスティングできます。この場合、S3 バケットをウェブサイトと同様に機能するよう設定できます。Amazon S3 上でウェブサイトをホスティングするステップを説明する例については、「チュートリアル: Amazon S3 での静的ウェブサイトの設定」を参照してください。

  パブリックアクセスをブロックするなどのセキュリティ設定を有効にして静的ウェブサイトをホストするには、Amazon CloudFront とオリジンアクセスコントロール (OAC) を使用し、HTTPS などの追加のセキュリティヘッダーを実装することをお勧めします。詳細については、「安全な静的ウェブサイトの使用開始」を参照してください。

  > **Note**
  >
  > Amazon S3 は、静的ウェブサイトアクセスの仮想ホスト形式とパス形式の URL の両方をサポートしています。バケットはパス形式の URL と仮想ホスト形式の URL を使用してアクセスできるため、DNS 準拠のバケット名を使用してバケットを作成することをお勧めします。詳細については、「バケットの制約と制限」を参照してください。

- 共有データセット — Amazon S3 で規模を拡大するにつれて、共有バケット内の固有のプレフィックスにさまざまなエンドカスタマーまたはビジネスユニットを割り当てるマルチテナントモデルを採用するのが一般的です。Amazon S3 アクセスポイントを使用すると、共有データセットにアクセスする必要のあるアプリケーションごとに、1 つの大きなバケットポリシーを個別のアクセスポイントポリシーに分割できます。このアプローチにより、共有データセット内で他のアプリケーションが実行中の操作を中断することなく、アプリケーションに適したアクセスポリシーの構築に集中しやすくなります。詳細については、「Amazon S3 アクセスポイントを使用したデータアクセスの管理」を参照してください。

- 高スループットワークロード - Mountpoint for Amazon S3 は、Amazon S3 バケットをローカルファイルシステムとしてマウントするための、高スループットのオープンソースファイルクライアントです。Mountpoint を使用すると、アプリケーションは、開く、読み取るなどのファイルシステム操作を通じて Amazon S3 に保存されているオブジェクトにアクセスできます。Mountpoint はこれらの操作を S3 オブジェクト API 呼び出しに自動的に変換し、アプリケーションが Amazon S3 のエラスティックなストレージとスループットにファイルインターフェイスを通じてアクセスできるようにします。詳細については、「Mountpoint for Amazon S3 の使用」を参照してください。
- マルチリージョンアプリケーション - Amazon S3 マルチリージョンアクセスポイントを使用すると、複数の AWS リージョン にある S3 バケットからのリクエストをアプリケーションが実行するために使用できるグローバルエンドポイントを作成できます。マルチリージョンアクセスポイントを使用して、単一のリージョンで使用するのと同じシンプルなアーキテクチャでマルチリージョンアプリケーションを構築し、世界中のどこでもこれらのアプリケーションを実行することができます。マルチリージョンのアクセスポイントは、パブリックインターネット経由でリクエストを送信する代わりに、Amazon S3 へのインターネットベースのリクエストを高速化する組み込みのネットワーク耐障害性を実現します。詳細については、「Amazon S3 マルチリージョンアクセスポイント」を参照してください。
- 新しいアプリケーションの構築 - Amazon S3 でのアプリケーション開発に SDK を使用できます。AWS SDK は、基盤となる Amazon S3 REST API をラップして、プログラミング作業を簡素化します。接続されるモバイルおよびウェブアプリケーションを構築するには、AWS Mobile SDK と AWS Amplify JavaScript ライブラリを使用することができます。詳細については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。
- Secure Shell (SSH) File Transfer プロトコル (SFTP) — インターネット経由で機密データを安全に転送する場合は、Amazon S3 バケットで SFTP 対応サーバーを使用できます。AWS SFTP は、SSH のセキュリティおよび認証機能をすべてサポートするプロトコルです。このプロトコルでは、ユーザー ID、権限、キーをきめ細かく制御したり、IAM ポリシーを使用してアクセスを管理したりできます。SFTP 対応サーバーを Amazon S3 バケットに関連付けるには、まず SFTP 対応サーバーを作成してください。次に、ユーザーアカウントを設定し、サーバーを Amazon S3 バケットに関連付けます。このプロセスのウォークスルーについては、「AWS ブログ」の「Amazon S3 用のフルマネージド SFTP サービス」を参照してください。

バケットの一覧表示

すべてのバケットを一覧表示するには、`s3:ListAllMyBuckets` 権限が必要です。バケットにアクセスするには、指定したバケットの内容を一覧表示するために必要な AWS Identity and Access Management (IAM) 権限も必ず取得してください。S3 バケットにアクセス権限を付与するバケット

ページ例については、「バケット の 1 つへのアクセスを IAM ユーザーに許可する」を参照してください。「HTTP アクセスが拒否されました (403 Forbidden)」エラーが発生する場合は、「バケットポリシーと IAM ポリシー」を参照してください。

バケットは、Amazon S3 コンソール、AWS CLI、または AWS SDK を使って一覧表示できます。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [汎用バケット] リストで、表示するバケットを選択します。

> **Note**
>
> [汎用バケット] リストには、すべての AWS リージョン に配置されているバケットが表示されます。

AWS CLI の使用

AWS CLI を使用して S3 バケットにアクセスしたり、S3 バケットのリストを生成したりするには、`ls` コマンドを使用します。バケット内のすべてのオブジェクトを一覧表示するときは、`s3:ListBucket` 権限が必要であることに注意してください。

このコマンド例を使用するには、*DOC-EXAMPLE-BUCKET1* をバケットの名前に置き換えます。

```
$ aws s3 ls s3://DOC-EXAMPLE-BUCKET1
```

次のコマンド例では、アカウントのすべての Amazon S3 バケットを一覧表示しています。

```
$ aws s3 ls
```

詳細および例については、「バケットおよびオブジェクトを一覧表示する」を参照してください。

AWS SDK の使用

ListBuckets API オペレーションを使用して、Amazon S3 バケットにアクセスすることもできます。このオペレーションをさまざまな AWS SDK で使用する方法の例については、「AWS SDK または CLI で ListBuckets を使用する」を参照してください。

バケットの作成

Amazon S3 にデータをアップロードするときは、いずれかの AWS リージョン に S3 バケットを作成しておく必要があります。バケットを作成するときは、バケット名とリージョンを選択します。必要に応じて、このバケットに他のストレージ管理オプションを選択できます。一度バケットを作成したら、そのバケット名またはリージョンを変更することはできません。バケットの命名についてさらに詳しくは、「バケットの名前付け」を参照してください。

バケットを作成する AWS アカウント は、そのバケットを所有します。バケットにはオブジェクトをいくつでもアップロードできます。デフォルトでは、AWS アカウント につき最大で 100 個のバケットを作成できます。バケットを追加する必要が場合は、サービスの制限の緩和を申請することによって、アカウントバケットの制限を最大 1,000 バケットまで引き上げることができます。バケットの上限緩和を申請する方法については、AWS 全般のリファレンスの「AWS のサービス クォータ」を参照してください。バケットには、オブジェクトをいくつでも保存できます。

S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、バケットにアップロードされる新しいオブジェクト所有権を制御し、アクセスコントロールリスト (ACL) を無効にするのに使用できます。デフォルトでは、オブジェクト所有権はバケット所有者の強制設定に設定され、すべての ACL は無効になります。ACL を無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、ポリシーのみを使用してデータへのアクセスを管理します。

詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

Amazon S3 マネージドキーによるサーバーサイド暗号化 (SSE-S3) は、Amazon S3 のすべてのバケットに対する暗号化設定の基本レベルです。S3 バケットにアップロードされたすべての新しいオブジェクトは、暗号化設定の基本レベルとして SSE-S3 を使用して自動的に暗号化されます。別の種類のデフォルトの暗号化を使用する場合は、AWS Key Management Service (AWS KMS) キー (SSE-KMS) またはお客様が用意したキー (SSE-C) を使用したサーバー側の暗号化を指定して、データを暗号化することもできます。詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。

バケットを作成するときは、Amazon S3 コンソール、Amazon S3 API、AWS CLI、または AWS SDK を使用できます。必要なアクセス権限の詳細については、Amazon Simple Storage Service API リファレンスの CreateBucket を参照してください。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ページ上部にあるナビゲーションバーで、現在表示されている AWS リージョン の名前をクリックします。次に、バケットを作成するリージョンを選択します。

   > **Note**
   >
   > レイテンシーとコストを最小化するため、さらに規制条件に対応するために、最寄りのリージョンを選択します。明示的に別のリージョンに移動する場合を除き、特定のリージョンに保管されたオブジェクトは、そのリージョンから移動されることはありません。Amazon S3 AWS リージョン のリストについては、「Amazon Web Services 全般のリファレンス」の「AWS のサービス エンドポイント」を参照してください。

3. 左側のナビゲーションペインで、[バケット] を選択します。
4. [Create bucket (バケットの作成)] を選択します。

   [バケットの作成] ページが開きます。
5. [全般設定] で、バケットが作成される AWS リージョン を確認します。
6. [バケットタイプ] で、[汎用] を選択します。
7. [バケット名] にバケットの名前を入力します。

   バケット名には次の条件があります。

   - パーティション内で一意にする必要があります。パーティションは、リージョンのグループです。AWS には、現在、aws (標準リージョン)、aws-cn (中国リージョン)、および aws-us-gov (AWS GovCloud (US) Regions) の 3 つのパーティションがあります。
   - 3 ~ 63 文字で指定する。
   - 小文字、数字、ドット (.)、およびハイフン (-) のみで構成できます。互換性を最も高くするには、静的ウェブサイトホスティング専用のバケットを除き、バケット名にドット (.) を使用しないことをお勧めします。
   - 文字や数字で始まり、文字や数字で終わります。

   バケットを作成したら、その名前を変更することはできません。バケットの命名の詳細については、「バケットの名前付け」を参照してください。

> **Important**
>
> バケット名にアカウント番号などの機密情報を含めないでください。バケット名は、バケット内のオブジェクトを参照する URL に表示されます。

8. AWS Management Console では、既存のバケットの設定を新しいバケットにコピーできます。既存のバケットの設定をコピーしない場合は、次のステップにスキップします。

> **Note**
>
> このオプションの特徴:
>
> - AWS CLI では使用できません。コンソールでのみ利用できます。
> - ディレクトリバケットには利用できません。
> - バケットポリシーは既存のバケットから新しいバケットにコピーしません。

既存のバケットの設定をコピーするには、[既存のバケットから設定をコピー] で [バケットを選択] をクリックします。[バケットを選択] ウィンドウが開きます。コピーする設定を持つバケットを検索して、[バケットを選択] をクリックします。[バケットを選択] ウィンドウが閉じて、[バケットを作成] ウィンドウが再び開きます。

[既存のバケットから設定をコピー] に、選択したバケットの名前が表示されるようになります。コピーしたバケット設定を削除するための [デフォルトを復元] オプションも表示されます。[バケットを作成] ページで、バケットの残りの設定を確認します。ここで、選択したバケットの設定と一致していることを確認できます。最後のステップにスキップできます。

9. [オブジェクト所有者] で、ACL を無効または有効にし、バケットにアップロードされたオブジェクトの所有権を制御するには、次のいずれかの設定を選択します。

ACL は無効です

- バケット所有者強制 (デフォルト) – ACL は無効になり、バケット所有者はバケット内のすべてのオブジェクトを自動的に所有し、完全に制御します。ACL は、S3 バケット内のデータに対するアクセス許可に対して影響を与えません。このバケットはアクセスコントロールを定義するためだけにポリシーを使用します。

デフォルトでは、ACL は無効になっています。Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

ACL の有効化

- 希望するバケット所有者 — バケット所有者は、他のアカウントが `bucket-owner-full-control` 既定 ACL でバケットに書き込む新しいオブジェクトを所有し、完全にコントロールできます。

  [バケット所有者を推奨] 設定を適用して、すべての Amazon S3 アップロードに `bucket-owner-full-control` 既定 ACL を含めることを要求する場合は、この ACL を使用するオブジェクトアップロードのみを許可するバケットポリシーを追加できます。
- オブジェクトライター — オブジェクトをアップロードする AWS アカウント は、そのオブジェクトを所有し、そのオブジェクトを完全にコントロールし、ACL を介して他のユーザーにそのオブジェクトへのアクセス権を付与できます。

> **Note**
>
> デフォルト設定は [バケット所有者の強制] です。デフォルト設定を適用して ACL を無効のままにするのに必要なのは、`s3:CreateBucket` アクセス許可のみです。ACL を有効にするためには、`s3:PutBucketOwnershipControls` アクセス許可が必要です。

10. [このバケットのパブリックアクセスブロック設定] で、バケットに適用するブロックパブリックアクセス設定を選択します。

    デフォルトでは、4 つすべての [パブリックアクセスをブロック] 設定が有効になっています。特定のユースケースでオフにする必要のある設定が 1 つ以上あることがわかっている場合を除き、すべての設定を有効にしておくことをお勧めします。ブロックパブリックアクセスの詳細については、Amazon S3 ストレージへのパブリックアクセスのブロック を参照してください。

> **Note**
>
> すべての [パブリックアクセスをブロック] 設定を有効にするのに必要なのは、`s3:CreateBucket` アクセス許可のみです。[パブリックアクセスをブロック] 設定をオフにするには、`s3:PutBucketPublicAccessBlock` アクセス許可が必要です。

11. (オプション) [バケットのバージョニング] では、オブジェクトのバリアントをバケットに保持するかどうかを選択できます。バージョニングの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

    バケットのバージョニングを無効または有効にするには、[Disable] (無効化) または [Enable] (有効化) を選択します。

12. (オプション) [Tags] (タグ) では、バケットにタグを追加することを選択できます。タグは、ストレージの分類に使用されるキーと値のペアです。

    (オプション) タグを追加するには、[キー] を入力してから、オプションの [値] を入力し、[タグの追加] を選択します。

13. [デフォルトの暗号化] で、[編集] を選択します。

14. デフォルトの暗号化を設定するには、[暗号化タイプ] で次のいずれかを選択します。

    - Amazon S3 マネージドキー (SSE-S3)
    - AWS Key Management Service キー (SSE-KMS)

    > **Important**
    >
    > デフォルト暗号化設定に SSE-KMS オプションを使用する場合、AWS KMS の 1 秒あたりのリクエスト (RPS) 制限が適用されます。AWS KMS クォータの詳細およびクォータの引き上げをリクエストする方法については、「AWS Key Management Service デベロッパーガイド」の「クォータ」を参照してください。

    バケットと新しいオブジェクトは、暗号化設定の基本レベルとして Amazon S3 マネージドキーを使用したサーバー側の暗号化で暗号化されます。デフォルトの暗号化の詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。

Amazon S3 のサーバー側の暗号化を使用してデータを暗号化する方法の詳細については、「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)」を参照してください。

15. [AWS Key Management Service キー (SSE-KMS)] を選択した場合は、以下の操作を実行します。

    a. [AWS KMS キー] で、次のいずれかの方法で KMS キーを指定します。

    - 使用可能な KMS キーのリストから選択するには、[AWS KMS keys から選択する] を選択し、使用可能なキーのリストから自分の KMS キーを選択します。

      AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がこのリストに表示されます。カスタマーマネージドキーの詳細については、AWS Key Management Service デベロッパーガイドの「カスタマーキーと AWS キー」を参照してください。

    - KMS キー ARN を入力するには、[AWS KMS key ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。
    - AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

      AWS KMS key の作成の詳細については、AWS Key Management Service デベロッパーガイドのキーの作成を参照してください。

> **Important**
>
> バケットと同じ AWS リージョン で使用可能な KMS キーのみを使用できます。Amazon S3 コンソールには、バケットと同じリージョンで最初の 100 個の KMS キーしか表示されません。リストに存在しない KMS キーを使用するには、KMS キー ARN を入力する必要があります。別のアカウントが所有している KMS キーを使用する場合は、まずそのキーを使用するアクセス許可が必要であり、次に KMS キー ARN を入力する必要があります。KMS キーのクロスアカウント権限の詳細については、AWS Key Management Service デベロッパーガイドの「他のアカウントで使用できる KMS キーを作成する」を参照してください。SSE-KMS に関する詳細は、「AWS KMS (SSE-KMS) によるサーバー側の暗号化の指定」を参照してください。
> Amazon S3 でサーバー側の暗号化に AWS KMS key を使用する場合は、対称暗号化 KMS キーを選択する必要があります。Amazon S3 では、対称暗号化 KMS キーのみがサポートされ、非対称暗号化 KMS キーはサポートされません。詳細

については、AWS Key Management Service デベロッパーガイドの「Identifying symmetric and asymmetric KMS keys」(対称および非対称 KMS キーの識別) を参照してください。

AWS KMS key の作成の詳細については、AWS Key Management Service デベロッパーガイドのキーの作成を参照してください。Amazon S3 での AWS KMS の使用に関する詳細は、「AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用」を参照してください。

b. SSE-KMS でデフォルトの暗号化を使用するようにバケットを設定する場合は、S3 バケットキーを有効にすることもできます。S3 バケットキーは、Amazon S3 から AWS KMS へのリクエストトラフィックを減らし、暗号化のコストを削減します。詳細については、「Amazon S3 バケットキーを使用した SSE-KMS のコストの削減」を参照してください。

   S3 バケットキーを使用するには、[バケットキー] で [有効化] を選択します。

16. (オプション) S3 オブジェクトロックを有効にする場合は、次の手順に従います。

a. [詳細設定] を選択します。

> **Important**
>
> バケットに対してオブジェクトロックを有効にすると、バージョニングも有効になります。有効にした後、オブジェクトロックのデフォルト保持設定およびリーガルホールド設定を指定し、新しいオブジェクトを削除または上書きしないようにする必要があります。

b. オブジェクトロックを有効にする場合は、[Enable] (有効化) を選択し、表示される警告を読んだうえで承認します。

詳細については、「S3 オブジェクトロックの使用」を参照してください。

> **Note**
>
> オブジェクトロックが有効なバケットを作成するには、`s3:CreateBucket`、`s3:PutBucketVersioning` および `s3:PutBucketObjectLockConfiguration` の許可が必要です。

17. [Create bucket (バケットの作成)] を選択します。

AWS SDK の使用

AWS SDK を使用してバケットを作成する場合は、最初にクライアントを作成し、次にそのクライアントを使用してバケットを作成するためのリクエストを送信します。ベストプラクティスとして、クライアントとバケットを同じ AWS リージョン で作成する必要があります。クライアントまたはバケットの作成時にリージョンを指定しない場合、Amazon S3 ではデフォルトのリージョンである米国東部 (バージニア北部) が使用されます。バケットの作成を特定の AWS リージョンに制限したい場合は、`LocationConstraint` 条件キーを使用します。

デュアルスタックのエンドポイントにアクセスするためにクライアントを作成するには、AWS リージョン を指定する必要があります。詳細については、「デュアルスタックのエンドポイント」を参照してください。使用可能な AWS リージョン リージョンのリストについては、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。

クライアントを作成すると、リージョンはリージョン固有のエンドポイントにマッピングされます。クライアントは、このエンドポイントを使用して Amazon S3 と通信します。`s3.region.amazonaws.com`。リージョンがローンチされたのが 2019 年 3 月 20 日以降である場合は、クライアントとバケットは同じリージョン内にある必要があります。2019 年 3 月 20 日以前にローンチされたリージョンの場合は、米国東部 (バージニア北部) のリージョンでクライアントを使用することによりバケットを作成できます。詳細については、「レガシーエンドポイント」を参照してください。

AWS SDK コード例が、以下のタスクを実行します。

- AWS リージョン を明示的に指定してクライアントを作成する - この例では、クライアントは `s3.us-west-2.amazonaws.com` エンドポイントを使用して Amazon S3 と通信します。任意の AWS リージョン を指定できます。AWS リージョン の一覧については、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。
- バケット名のみを指定してバケット作成リクエストを送信する - クライアントが Amazon S3 にリクエストを送信し、クライアントが作成されたリージョンにバケットを作成します。
- バケットの場所情報を取得する - Amazon S3 が、バケットの場所情報を、そのバケットに関連付けられている場所のサブリソースに保存します。

Java

この例では、AWS SDK for Java を使用して Amazon S3 バケットを作成する方法を示します。作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CreateBucketRequest;
import com.amazonaws.services.s3.model.GetBucketLocationRequest;

import java.io.IOException;

public class CreateBucket2 {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            if (!s3Client.doesBucketExistV2(bucketName)) {
                // Because the CreateBucketRequest object doesn't specify a region,
 the
                // bucket is created in the region specified in the client.
                s3Client.createBucket(new CreateBucketRequest(bucketName));

                // Verify that the bucket was created by retrieving it and checking
 its
                // location.
                String bucketLocation = s3Client.getBucketLocation(new
 GetBucketLocationRequest(bucketName));
                System.out.println("Bucket location: " + bucketLocation);
            }
```

```
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

作業サンプルの作成およびテストの手順については、「AWS SDK for .NET Version 3 API Reference」を参照してください。

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.S3.Util;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CreateBucketTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CreateBucketAsync().Wait();
        }

        static async Task CreateBucketAsync()
        {
```

```
            try
            {
                if (!(await AmazonS3Util.DoesS3BucketExistAsync(s3Client,
 bucketName)))
                {
                    var putBucketRequest = new PutBucketRequest
                    {
                        BucketName = bucketName,
                        UseClientRegion = true
                    };

                    PutBucketResponse putBucketResponse = await
 s3Client.PutBucketAsync(putBucketRequest);
                }
                // Retrieve the bucket location.
                string bucketLocation = await FindBucketLocationAsync(s3Client);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
        static async Task<string> FindBucketLocationAsync(IAmazonS3 client)
        {
            string bucketLocation;
            var request = new GetBucketLocationRequest()
            {
                BucketName = bucketName
            };
            GetBucketLocationResponse response = await
 client.GetBucketLocationAsync(request);
            bucketLocation = response.Location.ToString();
            return bucketLocation;
        }
    }
}
```

Ruby

作業サンプルの作成およびテストの手順については、「AWS SDK for Ruby - Version 3」を参照してください。

Example

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                  create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      "None. You must create a bucket before you can get its location!"
    else
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
```

```
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("doc-example-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

AWS CLI の使用

また、AWS Command Line Interface(AWS CLI) を使用して、S3 バケットを作成することもできます。詳細については、「AWS CLI コマンドリファレンス」の「create-bucket」を参照してください。

AWS CLI の詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS Command Line Interface とは」を参照してください。

S3 バケットのプロパティを表示するには

所有しているすべての Amazon S3 バケットのプロパティを表示できます。これらの設定には以下が含まれます。

- [バケットのバージョニング] – バージョニングを使用すると、1 つのバケットで複数バージョンのオブジェクトを維持できます。バージョニングは、新しいバケットに対してデフォルトで無効になります。バージョニングを有効にする詳細については、「バケットでのバージョニングの有効化」を参照してください。
- [Tags] (タグ) – AWS コスト配分では、バケットタグを使用してバケットの使用に対する請求に注釈を付けることができます。タグはキー/値ペアになっており、バケットに割り当てられるラベルを表します。詳細については、「S3 バケットタグでのコスト配分タグの使用」を参照してください。
- デフォルトの暗号化 – デフォルトの暗号化を有効にすると、サーバー側の自動暗号化が可能になります。Amazon S3 は、ディスクに保存する前にオブジェクトを暗号化し、ダウンロード時にオ

ブジェクトを復号化します。詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。

- サーバーアクセスのログ記録 – サーバーアクセスログ記録でバケットに対して行われたリクエストの詳細が記録されます。デフォルトでは、Amazon S3 によってサーバーアクセスログは収集されません。サーバーアクセスログ記録の有効化については、「Amazon S3 サーバーアクセスログを有効にします。」を参照してください。
- AWS CloudTrail データイベント – CloudTrail を使用してデータイベントをログに記録します。デフォルトでは、証跡はデータイベントを記録しません。追加の変更がイベントデータに適用されます。詳細については、AWS CloudTrail ユーザーガイドの証跡へのデータイベントのログ記録を参照してください。
- イベント通知 – イベントが発生するたびに送信先へ通知メッセージを送信する、Amazon S3 バケットイベントを有効にすることができます。詳細については、「Amazon S3 コンソールを使用したイベント通知の有効化と設定」を参照してください。
- Transfer Acceleration – クライアントと S3 バケットの間で、長距離にわたるファイル転送を高速、簡単、安全に行えるようになります。Transfer Acceleration の有効化の詳細については、「S3 Transfer Acceleration の有効化と使用」を参照してください。
- オブジェクトロック – S3 オブジェクトロックを使って、オブジェクトが固定期間または無期限に削除または上書きされることを防止できます。詳細については、「S3 オブジェクトロックの使用」を参照してください。
- リクエスタ支払い – バケット所有者ではなくリクエスタが、リクエストおよびデータ転送の料金の支払いを行う場合は、リクエスタ支払いを有効にします。詳細については、「ストレージ転送と使用量のリクエスタ支払いバケットの使用」を参照してください。
- 静的ウェブサイトホスティング – 静的ウェブサイトを Amazon S3 でホスティングできます。詳細については、「Amazon S3 を使用して静的ウェブサイトをホスティングする」を参照してください。

バケットプロパティは、AWS Management Console、AWS CLI、または AWS SDK を使用して表示できます。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、プロパティを表示するバケットの名前を選択します。
3. [プロパティ] タブを選択します。

4. [プロパティ] ページで、バケットの上記プロパティを設定できます。

AWS CLI の使用

AWS CLI を使用したバケットプロパティの表示

以下のコマンドは、AWS CLI を使用してさまざまなバケットプロパティを一覧表示する方法を示しています。

以下は、バケット *example-s3-bucket1* に関連付けられたタグセットを返します。バケットタグの詳細については、「S3 バケットタグでのコスト配分タグの使用」を参照してください。

```
aws s3api get-bucket-tagging --bucket example-s3-bucket1
```

詳細と例については、「AWS CLI コマンドリファレンス」の「get-bucket-tagging」を参照してください。

以下は、バケット *example-s3-bucket1* のバージョニング状態を返します。バケットのバージョニングの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

```
aws s3api get-bucket-versioning --bucket example-s3-bucket1
```

詳細と例については、「AWS CLI コマンドリファレンス」の「get-bucket-versioning」を参照してください。

以下は、バケット *example-s3-bucket1* のデフォルトの暗号化設定を返します。デフォルトでは、すべてのバケットには、Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) を使用するデフォルトの暗号化設定が指定されています。デフォルトのバケット暗号化の詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。

```
aws s3api get-bucket-encryption --bucket example-s3-bucket1
```

詳細と例については、「AWS CLI コマンドリファレンス」の「get-bucket-encryption」を参照してください。

以下は、バケット *example-s3-bucket1* の通知設定を返します。バケットイベント通知については、「Amazon S3 イベント通知」を参照してください。

```
aws s3api get-bucket-notification-configuration --bucket example-s3-bucket1
```

詳細と例については、「AWS CLI コマンドリファレンス」の「get-bucket-notification-configuration」を参照してください。

以下は、バケット *example-s3-bucket1* のログ記録ステータスを返します。バケットのログ記録については、「サーバーアクセスログによるリクエストのログ記録」を参照してください。

```
aws s3api get-bucket-logging --bucket example-s3-bucket1
```

詳細と例については、「AWS CLI コマンドリファレンス」の「get-bucket-logging」を参照してください。

AWS SDK の使用

バージョニング、タグなど、AWS SDKを使用してバケットを返す方法の例については、「AWS SDK を使用した Amazon S3 のアクション」を参照してください。

さまざまな AWS SDK の使用に関する一般的な情報については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。

バケットを空にする

バケットの中身は、Amazon S3 コンソール、AWS SDK、または AWS Command Line Interface (AWS CLI) を使って空にします。バケットを空にすると、すべてのオブジェクトが削除されますが、バケットは保持されます。バケットを空にすると、元に戻すことはできません。バケットを空にするアクションの実行中にバケットに追加されたオブジェクトは、削除される場合があります。バケット自体を削除する前に、バケット内のすべてのオブジェクト (すべてのオブジェクトバージョンと削除マーカーを含む) を削除する必要があります。

S3 バージョニングが有効化または一時停止されたバケットを空にすると、そのバケット内のすべてのオブジェクトの、すべてのバージョンが削除されます。詳細については、「バージョニングが有効なバケットでのオブジェクトの操作」を参照してください。

バケットのライフサイクル設定を指定してオブジェクトの有効期限を終了させ、Amazon S3 がそのオブジェクトを削除できるようにすることもできます。詳細については、「バケットにライフサイクル設定を設定する」を参照してください。大きなバケットを空にするには、S3 ライフサイクル設定ルールを使用することをお勧めします。ライフサイクルの有効期限は非同期プロセスであるため、バ

ケットが空になるまでのルールの実行には数日かかる場合があります。Amazon S3 が初めてルールを実行すると、有効期限の対象となるすべてのオブジェクトに削除対象のマークが付けられます。削除対象としてマークされたオブジェクトについては、請求が発生しなくなります。詳細については、「ライフサイクル設定ルールを使用して Amazon S3 バケットを空にするにはどうすればよいですか?」を参照してください。

S3 コンソールの使用

Amazon S3 コンソールを使用するとバケット内を空にできます。バケット内のすべてのオブジェクトを、バケットを削除することなく、削除できます。

S3 バケットを空にする

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット名] リストから、空にしたいバケットの名前の横にあるバケットアイコンを選び、[空にする] を選択します。
3. [バケットを空にする] ページで、テキストフィールドにバケット名を入力することでバケットを空にすることを確定し、[Empty (空にする)] を選択します。
4. [バケットを空にする: ステータス] ページで、バケットを空にするプロセスの進行状況をモニタリングします。

AWS CLI の使用

AWS CLI を使ってバケットを空にできるのは、バケットでバージョニングが有効化されていない場合のみです。バージョニングが有効化されていない場合は、AWS CLI コマンドの `rm` (削除) と `--recursive` パラメータを使って、バケットを空にすることができます (または、特定のキーネームプレフィックスを持つオブジェクトのサブセットを削除できます)。

次の `rm` コマンドは、たとえば doc や doc/doc1 といった、キーネームプレフィックス doc/doc2 を持つオブジェクトを削除します。

```
$ aws s3 rm s3://bucket-name/doc --recursive
```

プレフィックスを指定せずにすべてのオブジェクトを削除するには、次のコマンドを使用します。

```
$ aws s3 rm s3://bucket-name --recursive
```

詳細については、AWS Command Line Interface ユーザーガイドの「AWS CLI でのハイレベルな S3 コマンドの使用」を参照してください。

> **Note**
>
> バージョニングが有効化されたバケットから、オブジェクトを削除することはできません。Amazon S3 は、オブジェクトが削除される際に、このコマンドの目的である削除マーカーを追加します。S3 バケットのバージョニングの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

AWS SDK の使用

AWS SDK を使用すると、バケットを空にしたり、特定のキーネームプレフィックスを持つオブジェクトのサブセットを削除したりできます。

AWS SDK for Java を使ってバケットを空にする方法の例については、「バケットの削除」を参照してください。このコードは、バケットのバージョニングが有効化されてるか否かにかかわらず、すべてのオブジェクトを削除します。それによって、このバケットは削除されます。単にバケットを空にしたいときは、バケットを削除するステートメントが解除されていることを確認します。

その他の AWS SDK の使用方法についての詳細は、「アマゾン ウェブ サービスのツール」を参照してください。

ライフサイクル設定の使用

大きなバケットを空にするには、S3 ライフサイクル設定ルールを使用することをお勧めします。ライフサイクルの有効期限は非同期プロセスであるため、バケットが空になるまでのルールの実行には数日かかる場合があります。Amazon S3 が初めてルールを実行すると、有効期限の対象となるすべてのオブジェクトに削除対象のマークが付けられます。削除対象としてマークされたオブジェクトについては、請求が発生しなくなります。詳細については、「ライフサイクル設定ルールを使用して Amazon S3 バケットを空にするにはどうすればよいですか?」を参照してください。

ライフサイクル設定を使用してバケットを空にする場合、設定には、最新バージョン、最新でないバージョン、削除マーカー、および不完全なマルチパートアップロードが含まれている必要があります。

すべてのオブジェクト、または特定のキーネームプレフィックスが付いたオブジェクトのサブセットの期限が終了するようにライフサイクルの設定ルールを追加できます。たとえば、バケット内のすべ

てのオブジェクトを削除するには、作成後 1 日でオブジェクトが期限切れするようにライフサイクルのルールを設定できます。

Amazon S3 は、バケットライフサイクルルールがサポートしています。このルールを使用すると、開始後指定された日数内に完了しないマルチパートアップロードを停止できます。ストレージコストを最小限に抑えるように、このライフサイクルルールを設定することをお勧めします。詳細については、「不完全なマルチパートアップロードを削除するためのバケットライフサイクル設定の設定」を参照してください。

ライフサイクル設定を使用してバケットを空にする方法の詳細については、バケットにライフサイクル設定を設定する および オブジェクトの有効期限 を参照してください。

設定済みの AWS CloudTrail を持つバケットを空にする

AWS CloudTrail は、オブジェクトの削除など、Amazon S3 バケット内のオブジェクトレベルのデータイベントを追跡します。CloudTrail イベントの記録先としてバケットを使用し、同じバケットからオブジェクトを削除する場合は、バケットを空にするときに新しいオブジェクトを作成している可能性があります。これを防ぐには、AWS CloudTrail 証跡を中止します。CloudTrail 証跡がイベントをログに記録しないようにする方法の詳細については、「AWS CloudTrail ユーザーガイド」の「証跡のログ記録をオフにする」を参照してください。

CloudTrail 証跡がバケットに追加されないようにするもう 1 つの方法は、バケットポリシーに `s3:PutObject` ステートメントを追加することです。後で新しいオブジェクトをバケットに保存する場合は、この deny `s3:PutObject` ステートメントを削除する必要があります。詳細については、「オブジェクト操作」および「IAM ユーザーガイド」の「IAM JSON ポリシー要素: 効果」を参照してください。

バケットの削除

空の Amazon S3 バケットを削除できます。バケットを削除する前に、以下の点を考慮します。

- バケット名は一意です。バケットを削除すると、他の AWS ユーザーがその名前を使用できます。
- バケットが静的ウェブサイトをホストしていて、Amazon Route 53 ホストゾーンが チュートリアル: Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定 の説明通りに作成され設定されている場合は、バケットに関連する Route 53 ホストゾーンの設定をクリーンアップする必要があります。詳細については、「ステップ 2: Route 53 ホストゾーンを削除する」を参照してください。
- バケットに Elastic Load Balancing (ELB) から口グデータを配信している場合は、バケットを削除する前に、バケットへの ELB 口グの配信を停止することをお勧めします。バケットの削除後に、

別のユーザーが同じ名前でバケットを作成した場合、そのバケットにログデータが配信される可能性があります。ELB アクセスログの詳細については、Classic Load Balancer のユーザーガイドの「アクセスログ」、および Application Load Balancer のユーザーガイドの「アクセスログ」を参照してください。

トラブルシューティング

Amazon S3 バケットを削除できない場合は、次の点を考慮してください。

- バケットが空であることを確認します — バケット内にオブジェクトがないバケットのみ削除できます。バケットが空であることを確認します。
- アタッチされたアクセスポイントがないことを確認します – アタッチされたアクセスポイントがないバケットのみを削除することができます。バケットを削除する前に、バケットにアタッチされているアクセスポイントを削除します。
- AWS Organizationsサービスコントロールポリシー (SCP) — サービスコントロールポリシーは、バケットの削除許可を拒否できます。SCP の詳細については、AWS Organizations ユーザーガイドの「サービスコントロールポリシー」を参照してください。
- s3:DeleteBucket アクセス許可 – バケットを削除できない場合は、IAM 管理者と協力して、`s3:DeleteBucket` アクセス許可があることを確認してください。IAM アクセス許可を表示または更新する方法については、IAM ユーザーガイドの「IAM ユーザーの許可の変更」を参照してください。
- s3:DeleteBucket 拒否ステートメント – IAM ポリシーに `s3:DeleteBucket` アクセス権限があり、バケットを削除できない場合は、バケットポリシーに `s3:DeleteBucket` に対する拒否ステートメントが含まれている場合があります。ElasticBeanstalk によって作成されたバケットは、デフォルトでこのステートメントを含むポリシーを有しています。バケットを削除する前に、このステートメントまたはバケットポリシーを削除する必要があります。

> **Important**
>
> バケット名は一意です。バケットを削除すると、他の AWS ユーザーがその名前を使用できます。同じバケット名を使い続ける場合は、バケットを削除しないでください。バケットを空にして、それを維持することをお勧めします。

S3 コンソールの使用

S3 バケットを削除するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets (バケット名)] リストで、削除するバケットの名前の横にあるオプションを選択してから、ページの上部にある [Delete (削除)] を選択します。
3. [バケットを削除する] ページで、テキストフィールドにバケット名を入力することでバケットを削除することを確認し、[バケットを削除する] を選択します。

   > **Note**
   >
   > バケットにオブジェクトが含まれている場合は、[This bucket is not empty (このバケットは空ではありません)] というエラーアラートの [バケットを空にする設定] リンクを選択し、[バケットを空にする] ページの指示に従って、バケットを空にしてから削除します。次に、[バケットを削除する] ページに戻り、バケットを削除します。

4. バケットを削除したことを確認するには、[バケット] リストを開き、削除したバケットの名前を入力します。バケットが見つからない場合、削除は成功しています。

AWS SDK for Java の使用

次の例は、AWS SDK for Java を使ってバケットを削除する方法を示しています。まず、このコードはバケットのオブジェクトを削除し、そしてバケットを削除します。AWS SDK の詳細については、「アマゾン ウェブ サービスのツール」を参照してください。

Java

次の Java の例では、オブジェクトが含まれているバケットを削除します。すべてのオブジェクトを削除した後で、バケットを削除します。この例は、バージョニングの有無を問わず、使用できます。

> **Note**
>
> バージョニングが有効になっていないバケットの場合、すべてのオブジェクトを直接削除し、その後でバケットを削除できます。バージョニングが有効なバケットの場合は、バケットを削除する前に、すべてのオブジェクトバージョンを削除する必要があります。

作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.Iterator;

public class DeleteBucket2 {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Delete all objects from the bucket. This is sufficient
            // for unversioned buckets. For versioned buckets, when you attempt to
 delete
            // objects, Amazon S3 inserts
            // delete markers for all objects, but doesn't delete the object
 versions.
            // To delete objects from versioned buckets, delete all of the object
 versions
            // before deleting
```

```
            // the bucket (see below for an example).
            ObjectListing objectListing = s3Client.listObjects(bucketName);
            while (true) {
                Iterator<S3ObjectSummary> objIter =
objectListing.getObjectSummaries().iterator();
                while (objIter.hasNext()) {
                    s3Client.deleteObject(bucketName, objIter.next().getKey());
                }

                // If the bucket contains many objects, the listObjects() call
                // might not return all of the objects in the first listing. Check
to
                // see whether the listing was truncated. If so, retrieve the next
page of
                // objects
                // and delete them.
                if (objectListing.isTruncated()) {
                    objectListing = s3Client.listNextBatchOfObjects(objectListing);
                } else {
                    break;
                }
            }

            // Delete all object versions (required for versioned buckets).
            VersionListing versionList = s3Client.listVersions(new
ListVersionsRequest().withBucketName(bucketName));
            while (true) {
                Iterator<S3VersionSummary> versionIter =
versionList.getVersionSummaries().iterator();
                while (versionIter.hasNext()) {
                    S3VersionSummary vs = versionIter.next();
                    s3Client.deleteVersion(bucketName, vs.getKey(),
vs.getVersionId());
                }

                if (versionList.isTruncated()) {
                    versionList = s3Client.listNextBatchOfVersions(versionList);
                } else {
                    break;
                }
            }

            // After all objects and object versions are deleted, delete the bucket.
            s3Client.deleteBucket(bucketName);
```

```
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
 couldn't
            // parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

AWS CLI の使用

AWS CLI のオブジェクトを含むバケットは、バージョニングが有効になっていなければ削除できます。オブジェクトを含むバケットを削除すると、S3 Glacier ストレージクラスに移行済みのオブジェクトを含む、そのバケット内にあるすべてのオブジェクトが完全に削除されます。

バケットのバージョニングが有効になっていない場合は、rb (バケット削除) の AWS CLI コマンドを --force パラメータと共に使用することで、バケットとその内部のオブジェクトのすべてを削除できます。このコマンドは、すべてのオブジェクトを削除した後にバケットを削除します。

バージョニングが有効になっている場合、バージョニングされたオブジェクトはこのプロセスで削除されません。これにより、バケットが空にならないため、バケットの削除が失敗します。バージョニングされたオブジェクトの削除については、「オブジェクトバージョンの削除」を参照してください。

```
$ aws s3 rb s3://bucket-name --force
```

詳細については、AWS Command Line Interface ユーザーガイドの「AWS Command Line Interface でのハイレベルな S3 コマンドの使用」を参照してください。

Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

すべての Amazon S3 バケットにはデフォルトで暗号化が設定されており、オブジェクトは Amazon S3 マネージドキー (SSE-S3) を使用してサーバー側の暗号化により自動的に暗号化されます。この暗号化設定は Amazon S3 バケット内のすべてのオブジェクトに適用されます。

キーローテーションやアクセスポリシーの付与の管理など、キーをより細かく制御する必要がある場合は、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、または AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS) を使用できます。KMS キーの編集の詳細については、『AWS Key Management Service デベロッパーガイド』の「キーの編集」を参照してください。

> **Note**
>
> 新しいオブジェクトのアップロードを自動的に暗号化するようにバケットを変更しました。以前にデフォルトの暗号化を行わずにバケットを作成した場合、Amazon S3 は SSE-S3 を使用してバケットの暗号化をデフォルトで有効にします。SSE-S3 または SSE-KMS が設定されている既存のバケットについては、デフォルトの暗号化設定は変更されません。SSE-KMS でオブジェクトを暗号化する場合は、バケット設定で暗号化タイプを変更する必要があります。詳細については、「AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用」を参照してください。

SSE-KMS でデフォルトの暗号化を使用するようにバケットを設定する場合、S3 バケットキーを有効にして、Amazon S3 から AWS KMS へのリクエストトラフィックを減らし、暗号化のコストを削減することもできます。詳細については、「Amazon S3 バケットキーを使用した SSE-KMS のコストの削減」を参照してください。

デフォルトの暗号化で SSE-KMS が有効になっているバケットを特定するには、Amazon S3 ストレージレンズメトリクスを使用できます。S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージ分析機能です。詳細については、「Using S3 Storage Lens to protect your data」(S3 ストレージレンズを使用してデータを保護する) を参照してください。

サーバー側の暗号化を使用すると、Amazon S3 はオブジェクトをディスクに保存する前に暗号化し、オブジェクトをダウンロードするときに復号します。サーバー側の暗号化および暗号化キー管理を使用したデータ保護の詳細については、サーバー側の暗号化によるデータの保護 を参照してください。

デフォルトの暗号化に必要なアクセス許可の詳細については、『Amazon Simple Storage Service API リファレンス』の「PutBucketEncryption」を参照してください。

Amazon S3 コンソール、AWS SDK、Amazon S3 REST API、および AWS Command Line Interface (AWS CLI) を使用して、Amazon S3 で S3 バケットのデフォルトの暗号化動作を設定できます。

既存のオブジェクトの暗号化

暗号化されていない既存の Amazon S3 オブジェクトを暗号化するには、Amazon S3 バッチオペレーションを使用します。S3 バッチオペレーションは、操作するオブジェクトのリストとともに提供します。バッチオペレーションは各 API を呼び出して、指定されたオペレーションを実行します。バッチオペレーションのコピーオペレーションを使用して、既存の暗号化されていないオブジェクトをコピーし、同じバケットに新しい暗号化されたオブジェクトを書き込めます。1 つのバッチオペレーションジョブで、数十億ものオブジェクトに対して、指定されたオペレーションを実行できます。詳細については、Amazon S3 オブジェクトでの大規模なバッチ操作の実行 およびAWS ストレージブログの投稿である Amazon S3 バッチオペレーションによるオブジェクトの暗号化を参照してください。

`CopyObject` API オペレーションまたは `copy-object` AWS CLI コマンドを使用して、既存のオブジェクトを暗号化することもできます。詳細については、AWS ストレージブログの投稿である AWS CLI を使用した既存の Amazon S3 オブジェクトの暗号化を参照してください。

> **Note**
>
> デフォルトの暗号化が SSE-KMS に設定されている Amazon S3 バケットは、the section called “サーバーアクセスのログ記録” の送信先バケットとしては使用できません。サーバーアクセスログの送信先バケットは、デフォルトの暗号化として SSE-S3 のみをサポートしています。

クロスアカウント操作での SSE-KMS 暗号化の使用

クロスアカウント操作で暗号化を使用する場合は、次の点に注意してください。

- AWS KMS key Amazon リソースネーム (ARN) またはエイリアスがリクエスト時またはバケットのデフォルト暗号化設定によって提供されない場合は、AWS マネージドキー (aws/s3) が使用されます。
- KMS キーと同じ AWS アカウント の AWS Identity and Access Management (IAM) プリンシパルを使用して S3 オブジェクトをアップロードまたはアクセスする場合は、AWS マネージドキー (aws/s3) を使用できます。
- S3 オブジェクトにクロスアカウントアクセスを許可する場合は、カスタマーマネージドキーを使用します。カスタマーマネージドキーのポリシーを設定して、別のアカウントからのアクセスを許可することができます。
- カスタマーマネージド KMS キーを指定している場合、完全修飾 KMS キー ARN を使用することをお勧めします。代わりに KMS キーエイリアスを使用する場合、AWS KMS はリクエスタのアカウント内でキーを解決します。この動作により、バケット所有者ではなく、リクエスタに属する KMS キーでデータが暗号化される可能性があります。
- お客様 (リクエスタ) が Encrypt アクセス許可を付与されているキーを指定する必要があります。詳細については、AWS Key Management Service デベロッパーガイドの「キーユーザーが暗号化オペレーションに KMS キーを使用することを許可する」を参照してください。

カスタマーマネージドキーと AWS マネージド KMS キーをどのような場合に使用するかの詳細については、「Amazon S3 にあるオブジェクトの暗号化に AWS マネージドキー、またはカスタマーマネージドキーを使うべきですか」を参照してください。

レプリケーションでのデフォルト暗号化の使用

レプリケーション先バケットのデフォルトの暗号化を有効にすると、以下の暗号化動作が適用されます。

- レプリケート元バケットのオブジェクトが暗号化されていない場合、レプリケート先バケットのレプリカオブジェクトはレプリケート先バケットのデフォルトの暗号化設定を使用して暗号化されます。そのため、レプリケート元のオブジェクトのエンティティタグ (ETag) はレプリカオブジェクトの ETag とは異なります。アプリケーションで ETag を使用している場合は、アプリケーションを更新して、この違いを反映する必要があります。
- レプリケート元バケット内のオブジェクトが Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、または AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS) を使用して暗号化されている場合、レプリケート先バケットのレプリカオブジェクトは、レプリケート元オブジェクトと同じタイプの暗号化を使用します。レプリケート先バケットのデフォルトの暗号化設定は使用されません。

SSE-KMS を使用したデフォルト暗号化の使用については、「暗号化オブジェクトのレプリケート」を参照してください。

デフォルトの暗号化で Amazon S3 バケットキーを使用する

新しいオブジェクトで SSE-KMS をデフォルトの暗号化として使用するようにバケットを設定する場合は、S3 バケットキーを設定することもできます。S3 バケットキーは、Amazon S3 から AWS KMS へのトランザクションの数を減らし、SSE-KMS のコストを削減します。

新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するようにバケットを設定すると、AWS KMS によりバケットレベルのキーが生成されます。このキーは、バケット内のオブジェクトの一意のデータキーを作成するために使用されます。この S3 バケットキーは Amazon S3 内で期間限定で使用されるため、Amazon S3 で AWS KMS にリクエストを実行し、暗号化オペレーションを完了する必要性が軽減されます。

S3 バケットキーの使用の詳細については、Amazon S3 バケットキーの使用 を参照してください。

デフォルトの暗号化の設定

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための

自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

Amazon S3 バケットにはデフォルトで暗号化が有効化されており、新しいオブジェクトは Amazon S3 マネージドキー (SSE-S3) を使用してサーバー側の暗号化により自動的に暗号化されます。この暗号化は Amazon S3 バケット内のすべての新しいオブジェクトに適用され、費用はかかりません。

キーローテーションやアクセスポリシーの付与の管理など、暗号化キーをより細かく制御する必要がある場合は、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、または AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS) を使用できます。SSE-KMS に関する詳細は、「AWS KMS (SSE-KMS) によるサーバー側の暗号化の指定」を参照してください。DSSE-KMS に関する詳細については、「the section called “二層式サーバー側の暗号化 (DSSE-KMS) ”」を参照してください。

別のアカウントが所有している KMS キーを使用する場合は、そのキーを使用するアクセス許可が必要です。KMS キーのクロスアカウント権限の詳細については、「AWS Key Management Service デベロッパーガイド」の「他のアカウントで使用できる KMS キーを作成する」を参照してください。

デフォルトのバケット暗号化を SSE-KMS に設定すると、S3 バケットキーも設定して、AWS KMS リクエストのコストを削減することができます。詳細については、「Amazon S3 バケットキーを使用した SSE-KMS のコストの削減」を参照してください。

> **Note**
>
> PutBucketEncryption を使用してデフォルトのバケット暗号化を SSE-KMS に設定する場合、KMS キー ID が正しいことを確認する必要があります。Amazon S3 は PutBucketEncryption リクエストで提供された KMS キー ID を検証しません。

S3 バケットのデフォルトの暗号化の使用に追加料金はかかりません。デフォルトの暗号化動作を設定するためのリクエストには、標準 Amazon S3 リクエスト料金がかかります。料金については、Amazon S3 の料金を参照してください。SSE-KMS および DSSE-KMS の場合、AWS KMS の料金が適用され、これらは「AWS KMS の料金」に記載されています。

お客様が用意した暗号化キーを使用したサーバー側の暗号化 (SSE-C) はサポートされていません。

Amazon S3 コンソール、AWS SDK、Amazon S3 REST API、および AWS Command Line Interface (AWS CLI) を使用して、Amazon S3 で S3 バケットのデフォルトの暗号化を設定できます。

デフォルトの暗号化を有効にする前に注意する変更

バケットに対してデフォルトの暗号化を有効にした後は、以下の暗号化動作が適用されます。

- デフォルト暗号化が有効にされる前にバケットに存在していたオブジェクトの暗号化は、変更されません。
- デフォルト暗号化を有効にした後、オブジェクトをアップロードするとします。
  - PUT リクエストヘッダーに暗号化情報が含まれていない場合、Amazon S3 はオブジェクトを暗号化するために、バケットのデフォルトの暗号化設定を使用します。
  - PUT リクエストヘッダーに暗号化情報が含まれている場合、Amazon S3 は PUT リクエストの暗号化情報を使用して、オブジェクトを Amazon S3 に保存する前に暗号化します。
- デフォルト暗号化設定として SSE-KMS または DSSE-KMS オプションを使用する場合、AWS KMS の 1 秒あたりのリクエスト数 (RPS) 制限が適用されます。AWS KMS クォータの詳細およびクォータの引き上げをリクエストする方法については、「AWS Key Management Service デベロッパーガイド」の「クォータ」を参照してください。

> **Note**
>
> デフォルトの暗号化が有効になる前にアップロードされたオブジェクトは、暗号化されません。既存のオブジェクトの暗号化の詳細については、「the section called “デフォルトのバケット暗号化の設定”」を参照してください。

S3 コンソールの使用

Amazon S3 バケットにデフォルトの暗号化を設定するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、目的のバケットの名前を選択します。
4. [プロパティ] タブを選択します。
5. [デフォルトの暗号化] で、[編集] を選択します。

6. 暗号化を設定するには、[暗号化タイプ] で次のいずれかを選択します。

   - Amazon S3 マネージドキーを用いたサーバー側の暗号化 (SSE-S3)
   - AWS Key Management Service キーによるサーバー側の暗号化 (SSE-KMS)
   - AWS Key Management Service キーによる二層式サーバー側の暗号化 (DSSE-KMS)

   > **Important**
   >
   > デフォルトの暗号化設定として SSE-KMS または DSSE-KMS オプションを使用する場合、AWS KMS の 1 秒あたりのリクエスト数 (RPS) 制限が適用されます。AWS KMS クォータの詳細およびクォータの引き上げをリクエストする方法については、「AWS Key Management Service デベロッパーガイド」の「クォータ」を参照してください。

   バケットに別のタイプのデフォルト暗号化を指定しない限り、バケットと新しいオブジェクトはデフォルトで SSE-S3 で暗号化されます。デフォルトの暗号化の詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。

   Amazon S3 のサーバー側の暗号化を使用してデータを暗号化する方法の詳細については、「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)」を参照してください。

7. AWS Key Management Service キーによるサーバー側の暗号化 (SSE-KMS) または AWS Key Management Service キーによる二層式サーバー側の暗号化 (DSSE-KMS) を選択した場合は、次の操作を行います。

   a. [AWS KMS キー] で、次のいずれかの方法で KMS キーを指定します。

      - 使用可能な KMS キーのリストから選択するには、[AWS KMS keys から選択する] を選択し、使用可能なキーのリストから自分の KMS キーを選択します。

        AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がこのリストに表示されます。カスタマーマネージドキーの詳細については、AWS Key Management Service デベロッパーガイドの「カスタマーキーと AWS キー」を参照してください。
      - KMS キー ARN を入力するには、[AWS KMS key ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。
      - AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

AWS KMS key の作成の詳細については、AWS Key Management Service デベロッパーガイドのキーの作成を参照してください。

> **Important**
>
> バケットと同じ AWS リージョン で有効になっている KMS キーのみを使用できます。[Choose from your KMS keys] (KMS キーから選択する) を選択する場合、S3 コンソールにはリージョンごとに 100 個の KMS キーしか表示されません。同じリージョンに 100 個以上の KMS キーがある場合、S3 コンソールには最初の 100 個の KMS キーしか表示されません。コンソールに表示されていない KMS キーを使用するには、[AWS KMS key ARN を入力] を選択し、KMS キー ARN を入力します。Amazon S3 でサーバー側の暗号化に AWS KMS key を使用する場合は、対称暗号化 KMS キーを選択する必要があります。Amazon S3 は、対称暗号化 KMS キーのみをサポートします。このキーの詳細については、「AWS Key Management Service デベロッパーガイド」の「対称暗号化 KMS キー」を参照してください。

Amazon S3 における SSE-KMS の使用の詳細については、「AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用」を参照してください。DSSE-KMS の使用の詳細については、「the section called “二層式サーバー側の暗号化 (DSSE-KMS) ”」を参照してください。

b. SSE-KMS でデフォルトの暗号化を使用するようにバケットを設定する場合は、S3 バケットキーを有効にすることもできます。S3 バケットキーは、Amazon S3 から AWS KMS へのリクエストトラフィックを減らし、暗号化のコストを削減します。詳細については、「Amazon S3 バケットキーを使用した SSE-KMS のコストの削減」を参照してください。

   S3 バケットキーを使用するには、[バケットキー] で [有効化] を選択します。

> **Note**
>
> S3 バケットキーは DSSE-KMS ではサポートされていません。

8. [Save changes] (変更の保存) をクリックします。

AWS CLI の使用

以下の例では、SSE-S3 を使用するか、SSE-KMS と S3 バケット キーを使用して、デフォルトの暗号化を設定する方法を示します。

デフォルトの暗号化の詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。AWS CLI を使用してデフォルトの暗号化を設定する方法の詳細については、put-bucket-encryption を参照してください。

Example - SSE-S3 を使用したデフォルトの暗号化

この例では、Amazon S3 マネージドキーを使用したデフォルトのバケット暗号化を設定します。

```
aws s3api put-bucket-encryption --bucket example-s3-bucket --server-side-encryption-
configuration '{
    "Rules": [
        {
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }
    ]
}'
```

Example - S3 バケットキーを使用した SSE-KMS でのデフォルトの暗号化

この例では、S3 バケットキーを使用して SSE-KMS でデフォルトのバケット暗号化を設定します。

```
aws s3api put-bucket-encryption --bucket example-s3-bucket --server-side-encryption-
configuration '{
    "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

REST API の使用

REST API PutBucketEncryption オペレーションを使用して、デフォルト暗号化を有効にし、使用するサーバー側の暗号化のタイプ (SSE-S3、SSE-KMS、または DSSE-KMS) を設定します。

詳細については、「Amazon Simple Storage Service API リファレンス」の「PutBucketEncryption」を参照してください。

AWS CloudTrail および Amazon EventBridge によるデフォルト暗号化のモニタリング

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

AWS CloudTrail イベントを使用して、Amazon S3 バケットのデフォルトの暗号化設定リクエストを追跡できます。CloudTrail ログでは、以下の API イベント名が使用されます。

- `PutBucketEncryption`
- `GetBucketEncryption`
- `DeleteBucketEncryption`

これらの API コールの CloudTrail イベントと一致するように EventBridge ルールを作成することもできます。CloudTrail イベントの詳細については、[コンソールを使用してバケット内のオブジェクトのログ記録を有効にする] を参照してください。EventBridge イベントの詳細については、「AWS のサービス からのイベント」を参照してください。

オブジェクトレベルの Amazon S3 アクションに CloudTrail ログを使用して、Amazon S3 への PUT および POST リクエストを追跡できます。これらのアクションを使用すると、受信 PUT リクエスト

に暗号化ヘッダーがない場合に、デフォルトの暗号化を使用してオブジェクトが暗号化されているかどうかを確認できます。

デフォルトの暗号化設定を使用して Amazon S3 がオブジェクトを暗号化すると、ログには名前と値のペアとして、`"SSEApplied":"Default_SSE_S3"`、`"SSEApplied":"Default_SSE_KMS"`、または `"SSEApplied":"Default_DSSE_KMS"` フィールドの 1 つが含まれます。

PUT の暗号化ヘッダーを使用して Amazon S3 がオブジェクトを暗号化すると、ログには名前と値のペアとして、`"SSEApplied":"SSE_S3"`、`"SSEApplied":"SSE_KMS"`、`"SSEApplied":"DSSE_KMS"`、または `"SSEApplied":"SSE_C"` フィールドの 1 つが含まれます。

マルチパートアップロードについては、この情報は `InitiateMultipartUpload` API リクエストに含まれています。CloudTrail と CloudWatch の併用の詳細については、「Amazon S3 のモニタリング」を参照してください。

Mountpoint for Amazon S3 の使用

Mountpoint for Amazon S3 は、Amazon S3 バケットをローカルファイルシステムとしてマウントするための、高スループットのオープンソースファイルクライアントです。Mountpoint を使用すると、アプリケーションは、開く、読み取るなどのファイルシステム操作を通じて Amazon S3 に保存されているオブジェクトにアクセスできます。Mountpoint はこれらの操作を S3 オブジェクト API 呼び出しに自動的に変換し、アプリケーションが Amazon S3 のエラスティックなストレージとスループットにファイルインターフェイスを通じてアクセスできるようにします。

Mountpoint for Amazon S3 は、データレイク、機械学習トレーニング、画像レンダリング、自動運転車シミュレーション、抽出、変換、ロード (ETL) など、本番環境での大規模な読み取り負荷の高いアプリケーション用に一般提供されています。

Mountpoint は基本的なファイルシステム操作をサポートし、最大 5 TB のサイズのファイルを読み取ることができます。既存のファイルを一覧表示して読み取ったり、新しいファイルを作成したりできます。既存のファイルの変更やディレクトリの削除はできません。シンボリックリンクやファイルロックもサポートしていません。Mountpoint は、共有ファイルシステムの機能や POSIX 形式のアクセス許可のすべては必要としないが、大規模な S3 データセットの読み書きに Amazon S3 のエラスティックなスループットを必要とするアプリケーションに最適です。詳細については、GitHub の「Mountpoint ファイルシステムの動作」を参照してください。POSIX の完全サポートを必要とするワークロードには、Amazon FSx for Lustre とその S3 バケットのリンクのサポートが推奨されます。

Mountpoint for Amazon S3 は、Linux オペレーティングシステム専用です。Mountpoint を使用して、S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive、S3 Intelligent-Tiering Archive アクセス階層、および S3 Intelligent-Tiering Deep Archive アクセス階層を除くすべてのストレージクラスの S3 オブジェクトにアクセスできます。

トピック

Mountpoint のインストール

Mountpoint for Amazon S3 のビルド済みパッケージの場合、コマンドラインを使用してダウンロードしてインストールできます。Mountpoint をダウンロードしてインストールする手順は、使用している Linux オペレーティングシステムによって異なります。

トピック

RPM ベースのディストリビューション (Amazon Linux、Fedora、CentOS、RHEL)

1. 以下のダウンロード URL をアーキテクチャ用にコピーしてください。

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.rpm
   ```

2. Mountpoint for Amazon S3 パッケージをダウンロードします。*download-link* を、前のステップの適切なダウンロード URL に置き換えてください。

```
wget download-link
```

3. (オプション) ダウンロードしたファイルの整合性と信頼性を検証します。まず、アーキテクチャに適した署名 URL をコピーします。

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm.asc
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.rpm.asc
   ```

   次に、「Mountpoint for Amazon S3 パッケージの署名の検証」を参照してください。
4. 次のコマンドを使用して パッケージをインストールします。

   ```
   sudo yum install ./mount-s3.rpm
   ```
5. 次のコマンドを入力して、Mountpoint が正常にインストールされていることを確認します。

   ```
   mount-s3 --version
   ```

   次のような出力が表示されます:

   ```
   mount-s3 1.3.1
   ```

DEB ベースのディストリビューション (Debian、Ubuntu)

1. ダウンロード URL をアーキテクチャ用にコピーしてください。

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.deb
   ```

   ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.deb
```

2. Mountpoint for Amazon S3 パッケージをダウンロードします。*download-link* を、前のステップの適切なダウンロード URL に置き換えてください。

```
wget download-link
```

3. (オプション) ダウンロードしたファイルの整合性と信頼性を検証します。まず、アーキテクチャに署名 URL をコピーします。

   x86_64:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.deb.asc
```

   ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.deb.asc
```

   次に、「Mountpoint for Amazon S3 パッケージの署名の検証」を参照してください。

4. 次のコマンドを使用してパッケージをインストールします。

```
sudo apt-get install ./mount-s3.deb
```

5. 次のコマンドを実行して、Mountpoint for Amazon S3 が正常にインストールされていることを確認します。

```
mount-s3 --version
```

   次のような出力が表示されます:

```
mount-s3 1.3.1
```

他の Linux ディストリビューション

1. オペレーティングシステムのマニュアルを参照して、必須の FUSE と libfuse2 パッケージ。をインストールしてください。

2. ダウンロード URL をアーキテクチャ用にコピーしてください。

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.tar.gz
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.tar.gz
   ```

3. Mountpoint for Amazon S3 パッケージをダウンロードします。*download-link* を、前のステップの適切なダウンロード URL に置き換えてください。

   ```
   wget download-link
   ```

4. (オプション) ダウンロードしたファイルの整合性と信頼性を検証します。まず、アーキテクチャに署名 URL をコピーします。

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.tar.gz.asc
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.tar.gz.asc
   ```

   次に、「Mountpoint for Amazon S3 パッケージの署名の検証」を参照してください。

5. 次のコマンドを使用してパッケージをインストールします。

   ```
   sudo mkdir -p /opt/aws/mountpoint-s3 && sudo tar -C /opt/aws/mountpoint-s3 -xzf ./
   mount-s3.tar.gz
   ```

6. `mount-s3` バイナリを `PATH` 環境変数に追加します。`$HOME/.profile` ファイルに、次の行を追加します。

   ```
   export PATH=$PATH:/opt/aws/mountpoint-s3/bin
   ```

   `.profile` ファイルを保存し、次のコマンドを実行します。

```
source $HOME/.profile
```

7. 次のコマンドを実行して、Mountpoint for Amazon S3 が正常にインストールされていることを確認します。

```
mount-s3 --version
```

次のような出力が表示されます:

```
mount-s3 1.3.1
```

Mountpoint for Amazon S3 パッケージの署名を検証します。

1. GnuPG (gpg コマンド) をインストールします。ダウンロードした Mountpoint for Amazon S3 の信頼性と整合性を検証するために必要です。GnuPG は、Amazon Linux Amazon マシンイメージ (AMI) にはデフォルトでインストールされます。GnuPG をインストール後、ステップ 2 に進みます。
2. 以下のコマンドを実行して Mountpoint パブリックキーをダウンロードします。

```
wget https://s3.amazonaws.com/mountpoint-s3-release/public_keys/KEYS
```

3. 以下のコマンドを実行して、Mountpoint パブリックキーをキーリングにインポートします。

```
gpg --import KEYS
```

4. 次のコマンドを実行して、Mountpoint パブリックキーのフィンガープリントを検証します。

```
gpg --fingerprint mountpoint-s3@amazon.com
```

表示されたフィンガープリント文字列が以下と一致することを確認します。

```
673F E406 1506 BB46 9A0E  F857 BE39 7A52 B086 DA5A
```

フィンガープリント文字列が一致しない場合は、Mountpoint のインストールを完了せずに AWS Support に問い合わせてください。

5. パッケージ署名ファイルをダウンロードします。*signature-link* を、前のセクションの適切な署名リンクと置き換えます。

```
wget signature-link
```

6. 次のコマンドを実行して、ダウンロードしたパッケージの署名を検証します。*signature-filename* を、前のステップのファイル名に置き換えます。

```
gpg --verify signature-filename
```

例えば、RPM ベースのディストリビューション (Amazon Linux を含む) の場合は、次のコマンドを実行します。

```
gpg --verify mount-s3.rpm.asc
```

7. 出力にフレーズ `Good signature` が含まれている必要があります。出力にフレーズ `BAD signature` が含まれている場合、Mountpoint パッケージファイルを再ダウンロードし、上記の手順を繰り返します。問題が解決しない場合は、Mountpoint のインストールを完了せずに AWS Support に問い合わせてください。

出力には、信頼できる署名に関する警告が含まれる場合があります。これは問題を示すものではありません。これは、Mountpoint パブリックキーを独自に検証していないことを意味するだけです。

Mountpoint の設定と使用

Mountpoint for Amazon S3 を使用するには、ホストに AWS マウントしたい 1 つまたは複数のバケットにアクセスできる認証情報が必要です。さまざまな認証方法については、GitHub の「Mountpoint AWS 認証情報」を参照してください。

たとえば、新しい AWS Identity and Access Management (IAM) ユーザーとロールをこの目的を作成できます。このロールに、マウントしたい 1 つまたは複数のバケットへのアクセス権があることを確認してください。インスタンスプロファイルを使って、IAM ロールを Amazon EC2 インスタンスに渡すことができます。

Mountpoint for Amazon S3 の使用

Mountpoint for Amazon S3 を使用して次の操作を行います。

1. mount-s3 コマンドを使用してバケットをマウントします。

   以下の例では、*DOC-EXAMPLE-BUCKET* を S3 バケットの名前に置き換え、*~/mnt* を S3 バケットをマウントしたいホスト上のディレクトリに置き換えます。

   ```
   mkdir ~/mnt
   mount-s3 DOC-EXAMPLE-BUCKET ~/mnt
   ```

   デフォルトでは Mountpoint クライアントがバックグラウンドで実行されるため、*~/mnt* ディレクトリから S3 バケット内のオブジェクトにアクセスできるようになりました。

2. Mountpoint を通じてバケット内のオブジェクトにアクセスします。

   バケットをローカルにマウントしたら、cat や ls などの一般的な Linux コマンドして、S3 オブジェクトを操作することができます。Amazon S3 の Mountpoint は、S3 バケット内のキーをフォワードスラッシュ (/) 文字で分割することでファイルシステムパスとして解釈します。たとえば、Data/2023-01-01.csv バケットにオブジェクトキーがある場合、Mountpoint ファイルに Data という名前のディレクトリがあり、その中には 2023-01-01.csv という名前のファイルがあります。

   Mountpoint for Amazon S3 は、ファイルシステムに意図的に完全な POSIX 標準仕様を実装していません。Mountpoint は、ファイルシステムインターフェイスを通じて Amazon S3 に保存されているデータへの高スループットの読み取り/書き込みアクセスを必要しながらも、それ以外はファイルシステム機能に依存しないワークロード向けに最適化されています。詳細については、GitHub の「Mountpoint for Amazon S3 ファイルシステムの動作」を参照してください。より豊富なファイルシステムセマンティクスを必要とするお客様は、他の AWS ファイルサービス (例: Amazon Elastic File System (Amazon EFS) または Amazon FSx) などを検討してください。

3. umount コマンドを使用してバケットをアンマウントします。このコマンドは S3 バケットをアンマウントし、マウントポイントを終了します。

   以下のコマンド例を使用するには、*~/mnt* を S3 バケットがマウントされているホスト上のディレクトリに置き換えます。

   ```
   umount ~/mnt
   ```

> **Note**
>
> このコマンドのオプションのリストを取得するには、`umount --help` を実行します。

マウントポイント設定の詳細については、GitHub の「S3 バケット設定、およびファイルシステムの設定」を参照してください。

Mountpoint でのキャッシュの設定

Mountpoint for Amazon S3 を使用する場合、S3 バケットから最近アクセスされたデータを Amazon EC2 インスタンスストレージまたはアタッチされた Amazon EBS ボリュームにキャッシュするように設定できます。このようなデータをキャッシュすると、パフォーマンスの向上、データの繰り返しアクセスのコスト削減につながります。Mountpoint でのキャッシュは、複数の読み取り中に変更されない同じデータを繰り返し読み取るユースケースに最適です。たとえば、モデルの精度を向上させるためにトレーニングデータセットを複数回読み取る必要がある機械学習トレーニングジョブでキャッシュを使用できます。

S3 バケットをマウントする場合、オプションでフラグを使用してキャッシュを有効にできます。データキャッシュの場所とサイズ、メタデータをキャッシュに保持する期間を設定できます。バケットをマウントしてキャッシュを有効にすると、Mountpoint は設定したキャッシュの場所に空のサブディレクトリを作成します (このサブディレクトリがまだ存在しない場合)。最初にバケットをマウントする際、またマウントを解除する際、Mountpoint はキャッシュロケーションのコンテンツを削除します。Mountpoint でのキャッシュの設定と使用の詳細については、GitHub の「Mountpoint for Amazon S3 Caching configuration」を参照してください。

S3 バケットをマウントすると、`--cache` *`CACHE_PATH`* フラグを使用してキャッシュを有効にできます。次の例の、*`CACHE_PATH`* は、データをキャッシュするディレクトリへのファイルパスに置き換えます。*`DOC-EXAMPLE-BUCKET`* は S3 バケット名に置き換え、*`~/mnt`* を S3 バケットをマウントするホスト上のディレクトリに置き換えます。

```
mkdir ~/mnt
mount-s3 --cache CACHE_PATH DOC-EXAMPLE-BUCKET ~/mnt
```

> **Important**
>
> キャッシュを有効にすると、Mountpoint は、S3 バケットの暗号化されていないオブジェクトのコンテンツを、マウント時に設定したキャッシュの場所に保持します。データを保護するために、データキャッシュの場所へのアクセスを制限することをお勧めします。

Mountpoint のトラブルシューティング

Mountpoint for Amazon S3 は、AWS Support によってサポートされています。サポートが必要な場合は、AWS Support センターまでご連絡ください。

GitHub の Mountpoint 問題を確認して提出することもできます。

このプロジェクトで潜在的なセキュリティ上の問題を発見した場合、脆弱性報告ページから AWS セキュリティにその旨をお知らせください。GitHub で公開されている問題はご報告いただく必要はありません。

アプリケーションが Mountpoint で予期しない動作をする場合は、ログ情報を調べて問題を診断できます。

ログ記録

デフォルトでは、Mountpoint は重要度の高いログ情報をに syslog に出力します。

Amazon Linux を含む最新の Linux ディストリビューションを表示するには、`journald` コマンドを実行します。

```
journalctl -e SYSLOG_IDENTIFIER=mount-s3
```

その他の Linux システムでは、`syslog` エントリは `/var/log/syslog` のようなファイルに書き込まれる可能性があります

これらのログは、アプリケーションのトラブルシューティングに使用できます。たとえば、アプリケーションが既存のファイルを上書きしようとすると操作は失敗し、ログには次のような行が表示されます。

```
[WARN] open{req=12 ino=2}: mountpoint_s3::fuse: open failed: inode error: inode 2 (full
 key "README.md") is not writable
```

詳細については、GitHub の「Mountpoint for Amazon S3 ログ記録」を参照してください。

Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定

Amazon S3 Transfer Acceleration は、クライアントと S3 バケットの間で長距離にわたるファイル転送を高速、簡単、安全に行えるようにするバケットレベルの機能です。Transfer Acceleration は、世界各地から S3 バケットへの転送速度を最適化するように設計されています。Transfer Acceleration は、Amazon CloudFront の世界中に点在するエッジロケーションを利用します。エッジロケーションに到着したデータは、最適化されたネットワークパスで Amazon S3 にルーティングされます。

Transfer Acceleration を使用する場合、追加のデータ転送料金が発生することがあります。料金に関する詳細については、[Amazon S3 の料金] を参照してください。

なぜ Transfer acceleration を使用するのですか?

さまざまな理由で、バケットに対して Transfer Acceleration を使用することが推奨されます。

- 一元化されたバケットに世界中のお客様がアップロードします。
- 大陸間で定期的にギガバイトからテラバイト単位のデータを転送する。
- Amazon S3 にアップロードする際に、インターネット経由で利用可能な帯域幅をすべて使用することはできません。

Transfer Acceleration の使用が適している場合の詳細については、[Amazon S3 のよくある質問] を参照してください。

Transfer Acceleration を使用するための要件

S3 バケットで Transfer Acceleration を使用するための要件を以下に示します。

- Transfer Acceleration は仮想ホスト形式のリクエストでのみサポートされます。仮想ホスト形式のリクエストの詳細については、[REST API を使用したリクエストの実行] を参照してください。
- Transfer Acceleration で使用するバケットは、ピリオド (".") が含まれていない DNS 準拠の名前にする必要があります。
- Transfer Acceleration は、バケットに対して有効にする必要があります。詳細については、「S3 Transfer Acceleration の有効化と使用」を参照してください。

バケットで Transfer Acceleration を有効にした後、バケットへのデータ転送速度が上昇するまでに最大 20 分かかることがあります。

> **Note**
>
> 現在、Transfer Acceleration は以下のリージョンにあるバケットでサポートされています。
>
> - アジアパシフィック (東京) (ap-northeast-1)
> - アジアパシフィック (ソウル) (ap-northeast-2)
> - アジアパシフィック (ムンバイ) (ap-south-1)
> - アジアパシフィック (シンガポール) (ap-southeast-1)
> - アジアパシフィック (シドニー) (ap-southeast-2)
> - カナダ (中部) (ca-central-1)
> - ヨーロッパ (フランクフルト) (eu-central-1)
> - 欧州 (アイルランド) (eu-west-1)
> - ヨーロッパ (ロンドン) (eu-west-2)
> - 欧州 (パリ) (eu-west-3)
> - 南米 (サンパウロ) (sa-east-1)
> - 米国東部 (バージニア北部) (us-east-1)
> - 米国東部 (オハイオ) (us-east-2)
> - 米国西部 (北カリフォルニア) (us-west-1)
> - 米国西部 (オレゴン) (us-west-2)

- Transfer Acceleration が有効になっているバケットにアクセスするには、エンドポイント *bucketname*.s3-accelerate.amazonaws.com を使用する必要があります。または、デュアルスタックのエンドポイント *bucketname*.s3-accelerate.dualstack.amazonaws.com を使用して、IPv6 経由で有効なバケットに接続します。標準のデータ転送では、通常のエンドポイントを引き続き使用できます。
- 転送状態として高速化を設定するには、バケット所有者である必要があります。バケット所有者は他のユーザーに対し、バケットに高速化状態を設定するためのアクセス許可を割り当てることができます。s3:PutAccelerateConfiguration アクセス許可では、ユーザーがバケットで Transfer Acceleration を有効または無効にすることができます。ま

た、`s3:GetAccelerateConfiguration` アクセス許可では、ユーザーはバケットの Transfer Acceleration 状態 (`Enabled` または `Suspended`.) を返すことができます。

以下のセクションでは、Amazon S3 Transfer Acceleration の開始方法と、それを使用してデータを転送する方法について説明します。

トピック

- Amazon S3 Transfer Acceleration の開始方法
- S3 Transfer Acceleration の有効化と使用
- Amazon S3 Transfer Acceleration の速度比較ツールの使用

Amazon S3 Transfer Acceleration の開始方法

Amazon S3 Transfer Acceleration を使用すると、クライアントと S3 バケットの間で、長距離にわたるファイル転送を高速、簡単、安全に行えるようになります。Transfer Acceleration では、Amazon CloudFront の世界中に分散したエッジロケーションを使用します。エッジロケーションに到着したデータは、最適化されたネットワークパスで Amazon S3 にルーティングされます。

Amazon S3 Transfer Acceleration の使用を開始するには、以下の手順を実行します。

1. バケットで Transfer Acceleration を有効にする

   バケットでは、次のいずれかの方法で Transfer Acceleration を有効にすることができます。

   - Amazon S3 コンソールを使用します。
   - REST API の PUT Bucket accelerate オペレーションを使用する。
   - AWS CLI および AWS SDK を使用します。詳細については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。

   詳細については、「S3 Transfer Acceleration の有効化と使用」を参照してください。

   > **Note**
   >
   > バケットで Transfer Acceleration を使用するには、バケット名にピリオド (".") が含まれず、DNS 命名要件に準拠している必要があります。

2. 高速化を有効にしたバケットとの間でデータを転送する

次の s3-accelerate エンドポイントのドメイン名の 1 つを使用します。

- 高速化を有効にしたバケットにアクセスするには、*bucketname*.s3-accelerate.amazonaws.com を使用します。
- IPv6 経由で高速化を有効にしたバケットにアクセスするには、*bucketname*.s3-accelerate.dualstack.amazonaws.com を使用します。

  Amazon S3 デュアルスタックエンドポイントは、IPv6 および IPv4 を使用した S3 バケットへのリクエストをサポートしています。Transfer Acceleration デュアルスタックエンドポイントのみが、仮想ホスト形式のエンドポイント名を使用します。詳細については、IPv6 を使用したリクエストの実行の開始方法およびAmazon S3 デュアルスタックのエンドポイントの使用を参照してください。

> **Note**
>
> データ転送を高速化するには、データ転送アプリケーションがバケットにアクセスするために使用する 2 種類のエンドポイント (デュアルスタックエンドポイント用の .s3-accelerate.amazonaws.com または .s3-accelerate.dualstack.amazonaws.com) のどちらかを指定する必要があります。標準のデータ転送を使用する場合は、通常のエンドポイントを引き続き使用できます。

Transfer Acceleration を有効にしたあと、s3-accelerate エンドポイントドメイン名への Amazon S3 の PUT オブジェクトおよび GET オブジェクトのリクエストを指定できます。たとえば現在、PUT Object を使用する REST API アプリケーションで、PUT リクエストに mybucket.s3.us-east-1.amazonaws.com というホスト名を使用しているとします。PUT を高速化するには、リクエストのホスト名を mybucket.s3-accelerate.amazonaws.com に変更します。標準のアップロード速度に戻すには、名前を mybucket.s3.us-east-1.amazonaws.com に戻します。

Transfer Acceleration を有効にしたあと、パフォーマンス上の利点が感じられるまでには最長で 20 分ほどかかります。ただし、高速化エンドポイントは、Transfer Acceleration を有効にするとすぐに利用できます。

AWS CLI、AWS SDK、または他のツールで Amazon S3 との間でデータをやり取りする場合、高速化エンドポイントを使用して転送できます。AWS SDK を使用している場合、サポートされている一部の言語では高速化エンドポイントのクライアント設定フラグを使用できるため、Transfer Acceleration 用のエンドポイントを明示的に *bucketname*.s3-accelerate.amazonaws.com

に設定する必要はありません。高速化エンドポイントのクライアント設定フラグを使用する方法の例については、[S3 Transfer Acceleration の有効化と使用] を参照してください。

Transfer Acceleration エンドポイントでは、以下を除くすべての Amazon S3 オペレーションを使用できます。

- GET Service (バケットの一覧表示)
- PUT Bucket (バケットの作成)
- DELETE Bucket

また、Amazon S3 Transfer Acceleration は PUT Object - Copy を使用したリージョン間のコピーをサポートしていません。

S3 Transfer Acceleration の有効化と使用

Amazon S3 Transfer Acceleration を使用すると、クライアントと S3 バケットの間で、長距離にわたるファイル転送を迅速かつ安全に行うことができます。S3 コンソール、AWS Command Line Interface (AWS CLI)、API、または AWS SDK を使用して Transfer Acceleration を有効にすることができます。

このセクションでは、バケットに対する Amazon S3 Transfer Acceleration を有効にして、有効にしたバケットに高速化エンドポイントを使用する方法の例を示します。

Transfer Acceleration の要件の詳細については、[Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定] を参照してください。

S3 コンソールの使用

> **Note**
>
> 高速アップロード速度と非高速アップロード速度を比較する場合は、Amazon S3 Transfer Acceleration 速度比較ツールを開きます。
> 速度比較ツールでは、マルチパートアップロードを使用して、ブラウザからさまざまな AWS リージョンにファイルを転送し、Amazon S3 Transfer Acceleration を使用した場合と使用していない場合の速度を比較します。リージョン別に、ダイレクトアップロードの速度と、高速化アップロードの速度を比較できます。

S3 バケットの Transfer Acceleration を有効にするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、Transfer Acceleration を有効にするバケットの名前を選択します。
3. [プロパティ] を選択します。
4. [Transfer Acceleration] で、[編集] を選択します。
5. [有効化] を選択し、[変更を保存] を選択します。

高速データ転送にアクセスするには

1. Amazon S3 によってバケットの Transfer Acceleration が有効になると、バケットの プロパティ タブが表示されます。
2. Transfer acceleration の 高速化エンドポイント には、バケットの Transfer Acceleration エンドポイントが表示されます。このエンドポイントを使用して、バケットとの間の高速データ転送にアクセスします。

   Transfer Acceleration を停止した場合、高速化エンドポイントは機能しなくなります。

AWS CLI の使用

以下は、Transfer Acceleration で使用される AWS CLI コマンドの例です。AWS CLI をセットアップする手順については、[AWS CLI を使用した Amazon S3 での開発] を参照してください。

バケットでの Transfer Acceleration の有効化

AWS CLI put-bucket-accelerate-configuration コマンドを使用すると、バケットで Transfer Acceleration を有効化または停止できます。

次の例では、`Status=Enabled` を設定することにより、バケットに対する Transfer Acceleration を有効にしています。`Status=Suspended` を使用して、Transfer Acceleration を一時停止します。

Example

```
$ aws s3api put-bucket-accelerate-configuration --bucket bucketname --accelerate-configuration Status=Enabled
```

Transfer Acceleration の使用

S3 および s3api AWS CLI コマンドによって行われたすべての Amazon S3 リクエストを、高速化エンドポイント: `s3-accelerate.amazonaws.com` に送信できます。これを行うには、AWS Config ファイルのプロファイルで、設定値 `use_accelerate_endpoint` を `true` に設定します。高速化エンドポイントを使用するには、バケットに対する Transfer Acceleration を有効にしておく必要があります。

すべてのリクエストはバケットアドレス指定の仮想スタイル (`my-bucket.s3-accelerate.amazonaws.com`) を使用して送信されます。高速化エンドポイントでは、これらのオペレーションをサポートしていないため、`ListBuckets`、`CreateBucket`、`DeleteBucket` リクエストは高速化エンドポイントには送信されません。

`use_accelerate_endpoint` の詳細については、AWS CLI CLI コマンドリファレンスの「AWS CLI S3 の設定」を参照してください。

次の例では、デフォルトのプロファイル内で `use_accelerate_endpoint` を `true` に設定しています。

Example

```
$ aws configure set default.s3.use_accelerate_endpoint true
```

一部の AWS CLI コマンドについてのみ高速化エンドポイントを使用するには、次のいずれかの方法を使用できます。

- すべての s3 または s3api コマンドで高速化エンドポイントを使用するには、`--endpoint-url` パラメータを `https://s3-accelerate.amazonaws.com` に設定します。
- AWS Config ファイル内で別々のプロファイルを設定できます。たとえば、`use_accelerate_endpoint` を `true` に設定するプロファイルと `use_accelerate_endpoint` を設定しないプロファイルを作成します。コマンドを実行する際は、高速化エンドポイントを使用するかどうかによって、適切なプロファイルを指定します。

Transfer Acceleration が有効になっているバケットにオブジェクトをアップロードする

次の例では、高速化エンドポイントの使用が設定されているデフォルトプロファイルを使用して、Transfer Acceleration が有効になっているバケットにファイルをアップロードします。

Example

```
$ aws s3 cp file.txt s3://bucketname/keyname --region region
```

次の例では、--endpoint-url パラメータで高速化エンドポイントを指定して、Transfer Acceleration が有効になっているバケットにファイルをアップロードします。

Example

```
$ aws configure set s3.addressing_style virtual
$ aws s3 cp file.txt s3://bucketname/keyname --region region --endpoint-url https://s3-accelerate.amazonaws.com
```

AWS SDK の使用

以下は、AWS SDK で Transfer Acceleration を使用して Amazon S3 にオブジェクトをアップロードする例です。AWS SDK でサポートされている一部の言語 (Java、.NET など) では高速化エンドポイントのクライアント設定フラグを使用できるため、Transfer Acceleration 用のエンドポイントを明示的に *bucketname*.s3-accelerate.amazonaws.com に設定する必要はありません。

Java

Example

次の例では、高速化エンドポイントを使用して、オブジェクトを Amazon S3 にアップロードする方法を示しています。この例では、次のような処理を実行します。

- 高速化エンドポイントを使用するために設定される AmazonS3Client を作成します。クライアントがアクセスするすべてのバケットで、Transfer Acceleration が有効になっている必要があります。
- 指定したバケットで Transfer Acceleration を有効にします。このステップは、指定したバケットで Transfer Acceleration が有効になっていない場合に限り必要です。
- 指定されたバケットで Transfer Acceleration が有効になっていることを確認します。
- バケットの高速化エンドポイントを使用して、指定されたバケットに新しいオブジェクトをアップロードします。

Transfer Acceleration の使用方法については、「Amazon S3 Transfer Acceleration の開始方法」を参照してください。作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketAccelerateConfiguration;
import com.amazonaws.services.s3.model.BucketAccelerateStatus;
import com.amazonaws.services.s3.model.GetBucketAccelerateConfigurationRequest;
import com.amazonaws.services.s3.model.SetBucketAccelerateConfigurationRequest;

public class TransferAcceleration {
    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";

        try {
            // Create an Amazon S3 client that is configured to use the accelerate
 endpoint.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .enableAccelerateMode()
                    .build();

            // Enable Transfer Acceleration for the specified bucket.
            s3Client.setBucketAccelerateConfiguration(
                    new SetBucketAccelerateConfigurationRequest(bucketName,
                            new BucketAccelerateConfiguration(
                                    BucketAccelerateStatus.Enabled)));

            // Verify that transfer acceleration is enabled for the bucket.
            String accelerateStatus = s3Client.getBucketAccelerateConfiguration(
                    new GetBucketAccelerateConfigurationRequest(bucketName))
                    .getStatus();
            System.out.println("Bucket accelerate status: " + accelerateStatus);

            // Upload a new object using the accelerate endpoint.
            s3Client.putObject(bucketName, keyName, "Test object for transfer
 acceleration");
```

```
            System.out.println("Object \"" + keyName + "\" uploaded with transfer
 acceleration.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

次の例では、AWS SDK for .NET を設定することにより、バケットに対する Transfer Acceleration を有効にする方法を示しています。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TransferAccelerationTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
```

```
            EnableAccelerationAsync().Wait();
        }

        static async Task EnableAccelerationAsync()
        {
                try
                {
                    var putRequest = new PutBucketAccelerateConfigurationRequest
                    {
                        BucketName = bucketName,
                        AccelerateConfiguration = new AccelerateConfiguration
                        {
                            Status = BucketAccelerateStatus.Enabled
                        }
                    };
                    await
 s3Client.PutBucketAccelerateConfigurationAsync(putRequest);

                    var getRequest = new GetBucketAccelerateConfigurationRequest
                    {
                        BucketName = bucketName
                    };
                    var response = await
 s3Client.GetBucketAccelerateConfigurationAsync(getRequest);

                    Console.WriteLine("Acceleration state = '{0}' ",
 response.Status);
                }
                catch (AmazonS3Exception amazonS3Exception)
                {
                    Console.WriteLine(
                        "Error occurred. Message:'{0}' when setting transfer
 acceleration",
                        amazonS3Exception.Message);
                }
        }
    }
}
```

Transfer Acceleration が有効になっているバケットにオブジェクトをアップロードする際は、クライアントの作成時に高速化エンドポイントの使用を指定します。

```
var client = new AmazonS3Client(new AmazonS3Config
```

```
            {
                RegionEndpoint = TestRegionEndpoint,
                UseAccelerateEndpoint = true
            }
```

Javascript

AWS SDK for JavaScript を使用して Transfer Acceleration を有効にする例については、AWS SDK for JavaScript API リファレンスの「putBucketAccelerateConfiguration オペレーションの呼び出し」を参照してください。

Python (Boto)

SDK for Python を使用して Transfer Acceleration を有効にする例については、AWS SDK for Python (Boto3) API リファレンスの「put_bucket_accelerate_configuration」を参照してください。

Other

他の AWS SDK の使用については、サンプルコードとライブラリのページを参照してください。

REST API の使用

REST API `PutBucketAccelerateConfiguration` オペレーションを使用して、既存のバケットでアクセラレーション設定を有効にします。

詳細については、Amazon Simple Storage Service API リファレンスの「PutBucketAccelerateConfiguration」を参照してください。

Amazon S3 Transfer Acceleration の速度比較ツールの使用

Amazon S3 Transfer Acceleration の速度比較ツールを使用すると、高速化した場合と高速化していない場合の Amazon S3 リージョン間でのアップロード速度を比較できます。速度比較ツールでは、マルチパートアップロードを使用して、ブラウザからさまざまな Amazon S3 リージョンへのファイル転送を行い、Transfer Acceleration を使用した場合と使用していない場合の比較が行われます。

速度比較ツールには、次のいずれかの方法でアクセスできます。

- 下に示す URL をブラウザウィンドウにコピーします。このとき、*region* は使用している AWS リージョン (us-west-2 など) に、*yourBucketName* は評価するバケット名に置き換えてください。

```
https://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/
accelerate-speed-comparsion.html?
region=region&origBucketName=yourBucketName
```

Amazon S3 でサポートされているリージョンのリストについては、「AWS 全般のリファレンス」の「Amazon S3 エンドポイントとクォータ」を参照してください。

- Amazon S3 コンソールを使用します。

ストレージ転送と使用量のリクエスタ支払いバケットの使用

一般に、Amazon S3 のバケットのストレージおよびデータ転送にかかるコストはすべて、そのバケット所有者が負担します。ただし、バケットをリクエスタ支払いバケットに設定することはできます。リクエスタ支払いバケットの場合、リクエストおよびバケットからのデータのダウンロードにかかるコストは、所有者でなくリクエストを実行したリクエスタが支払います。データの保管にかかるコストは常にバケット所有者が支払います。

通常は、データを共有したいが、他者がデータにアクセスする際に発生する費用を負担したくない場合に、リクエスタ支払いバケットをバケットに設定します。例えば、郵便番号リスト、参照データ、地理空間情報、ウェブクロールデータといった、大規模なデータセットを利用できるようにする際にリクエスタ支払いバケットを設定することができます。

> **Important**
>
> バケットでリクエスタ支払いを有効化すると、そのバケットには匿名アクセスができなくなります。

リクエスタ支払いバケットに関するリクエストはすべて認証する必要があります。リクエストの認証により、Amazon S3 はリクエスタを特定し、リクエスタ支払いバケットの使用に対して課金できるようになります。

リクエスタがリクエストを送る前に AWS Identity and Access Management (IAM) ロールを引き受ける際に、ロールが属するアカウントにリクエストの料金が発生します。IAM ロールの詳細については、「IAM ユーザーガイド」の「IAM ロール」を参照してください。

バケットをリクエスタ支払いバケットとして設定した後、リクエスタはリクエストとデータダウンロードに対して課金されることを理解している旨を示す必要があります。料金を受け入れることを示

すには、リクエスタが DELETE、GET、HEAD、POST、PUT リクエストの API リクエストにヘッダーとして x-amz-request-payer を含めるか、REST リクエストに RequestPayer パラメータを追加する必要があります。CLI リクエストの場合、リクエスタは --request-payer パラメータを使用できます。

Example – オブジェクトの削除時にリクエスタ支払いを使用する

次の DeleteObjectVersion API の例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
DELETE /Key+?versionId=VersionId HTTP/1.1
Host: Bucket.s3.amazonaws.com
x-amz-mfa: MFA
x-amz-request-payer: RequestPayer
x-amz-bypass-governance-retention: BypassGovernanceRetention
x-amz-expected-bucket-owner: ExpectedBucketOwner
```

リクエスタが RestoreObject API を使用してオブジェクトを復元する場合、x-amz-request-payer ヘッダーまたは RequestPayer パラメータがリクエストに含まれている限りリクエスタ支払いがサポートされます。ただし、リクエスタはリクエストのコストのみを支払います。バケット所有者が取り出し料金を支払います。

リクエスタ支払いバケットは、次の場合サポートしていません。

- 匿名リクエスト
- SOAP リクエスト
- リクエスタ支払いバケットをエンドユーザーログのターゲットバケットとして使用する場合、あるいはその逆。ただし、ターゲットバケットがリクエスタ支払いバケットでない場合は、リクエスタ支払いバケット上でエンドユーザーのログを有効にできます。

リクエスタ支払いの課金の仕組み

成功したリクエスタ支払いリクエストに対する課金はシンプルです。リクエスタは、転送されたデータとリクエストに対して支払い、バケット所有者はデータのストレージに対して支払います。ただし以下の条件では、リクエストに対してバケット所有者が課金されます。

- このリクエストにより AccessDenied (HTTP 403 Forbidden) エラーが返され、リクエストがバケット所有者の個別の AWS アカウントまたは AWS 組織内で開始される。

- リクエストが SOAP リクエストの場合

リクエスタ支払いの詳細については、以下のトピックを参照してください。

トピック

- バケットでのリクエスタ支払いの設定
- REST API を使用した requestPayment 設定の取得
- リクエスタ支払いバケットからのオブジェクトのダウンロード

バケットでのリクエスタ支払いの設定

Amazon S3 バケットをリクエスタ支払いバケットに設定して、バケット所有者の代わりにリクエスタがリクエストとデータダウンロードの費用を支払うようにすることができます。

このセクションでは、コンソールと REST API を使用して Amazon S3 バケットでリクエスタ支払いを設定する方法の例を示します。

S3 コンソールの使用

S3 バケットのリクエスタ支払いを有効にするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] のリストで、リクエスタ支払いを有効にするバケットの名前を選択します。
3. [プロパティ] を選択します。
4. [リクエスタ支払い] で、[編集] を選択します。
5. [有効化] を選択し、[変更を保存] を選択します。

   Amazon S3 は、[バケットのリクエスタ支払い] を有効にし、[バケットの概要] を表示します。[リクエスタ支払い] の下に、[有効] と表示されます

REST API の使用

バケット所有者のみが、バケットの`RequestPaymentConfiguration.payer`の設定値を`BucketOwner` (デフォルト) または`Requester`に設定できます。`requestPayment` リソースの設定はオプションです。デフォルトでは、バケットはリクエスタ支払いバケットではありません。

リクエスタ支払いバケットを通常のバケットに戻すには、値 BucketOwner を使用します。通常、Amazon S3 バケットにデータをアップロードする際に BucketOwner を使用し、その後、バケット内のオブジェクトを公開する前に値を Requester に設定します。

requestPayment を設定するには

- PUT リクエストを使用して、指定したバケットの Payer の値を Requester に設定します。

```
PUT ?requestPayment HTTP/1.1
Host: [BucketName].s3.amazonaws.com
Content-Length: 173
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]

<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

リクエストが成功すると、Amazon S3 は、以下のようなレスポンスを返します。

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
x-amz-request-charged:requester
```

リクエスタ支払いは、バケットレベルでのみ設定できます。バケット内の特定のオブジェクトに対してリクエスタ支払いを設定することはできません。

バケットは、いつでも自由に BucketOwner または Requester に設定できます。ただし、新しい設定値が有効になるまでに数分かかる場合があります。

> **Note**
>
> 特に URL の有効期限がとても長い場合、署名付き URL を提供するバケット所有者は、バケットをリクエスタ支払いに設定する前によく検討してください。バケット所有者は、バ

ケット所有者の認証情報を使用する署名付き URL をリクエスタが使用するたびに課金されます。

REST API を使用した requestPayment 設定の取得

バケットに設定された Payer 値を取得するには、リソース requestPayment をリクエストします。

requestPayment リソースを返すには

- 以下のリクエストに示すように、GET リクエストを使用して requestPayment リソースを取得します。

```
GET ?requestPayment HTTP/1.1
Host: [BucketName].s3.amazonaws.com
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

リクエストが成功すると、Amazon S3 は、以下のようなレスポンスを返します。

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Type: [type]
Content-Length: [length]
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

このレスポンスでは、payer 値が Requester に設定されていることが示されています。

リクエスタ支払いバケットからのオブジェクトのダウンロード

リクエスタ支払いバケットからデータをダウンロードするとリクエスタに対して課金されるため、リクエストには、リクエスタが課金について認識していることを示すパラメータ x-amz-request-payer を含める必要があります。リクエスタ支払いバケット内のオブジェクトにアクセスするには、リクエストに以下のいずれかを含める必要があります。

- DELETE、GET、HEAD、POST、PUT リクエストの場合は、ヘッダーに x-amz-request-payer : requester を含めます
- 署名付き URL の場合は、リクエストに x-amz-request-payer=requester を含めます

リクエストが成功してリクエスタが課金されると、レスポンスには x-amz-request-charged:requester ヘッダーが含まれます。リクエスト内に x-amz-request-payer がない場合、Amazon S3 は 403 エラーを返し、リクエストに対してバケット所有者が課金されます。

> **Note**
>
> バケット所有者は、リクエストに x-amz-request-payer を含める必要はありません。署名計算内に x-amz-request-payer とその値を含めたことを確認してください。詳細については、「CanonicalizedAmzHeaders 要素の作成」を参照してください。

REST API の使用

リクエスタ支払いバケットからオブジェクトをダウンロードするには

- リクエスタ支払いバケットからオブジェクトをダウンロードするには、以下のリクエストに示すように、GET リクエストを使用します。

```
GET / [destinationObject] HTTP/1.1
Host: [BucketName].s3.amazonaws.com
x-amz-request-payer : requester
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

GET リクエストが成功してリクエスタが課金されると、レスポンスには x-amz-request-charged:requester が含まれます。

Amazon S3 は、リクエスタ支払いバケットからオブジェクトを取得するリクエストに対して `Access Denied` エラーを返すことがあります。詳細については、Amazon Simple Storage Service API リファレンスの Error Responses をご参照ください。

AWS CLI の使用

AWS CLI を使用してリクエスタ支払いバケットからオブジェクトをダウンロードするには、`get-object` リクエストの一部として `--request-payer requester` を指定します。詳細については、AWS CLI リファレンスの get-object を参照してください。

バケットの制約と制限

Amazon S3 の各バケットは、それを作成した AWS アカウント によって所有されます。バケットの所有権を別のアカウントに譲渡することはできません。

バケットの作成時にバケットの名前と作成先の AWS リージョン を選択します。バケットの作成後はバケットの名前もリージョンも変更できません。

バケットに名前を付けるときは、ユーザーまたはユーザーのビジネスに関連する名前を選択します。他のユーザーに関連付けられている名前は使用しないでください。例えば、バケット名に `AWS` や `Amazon` を使用しないでください。

デフォルトでは、AWS アカウント につき最大で 100 個のバケットを作成できます。追加のバケットが必要な場合は、クォータ引き上げリクエストを申請することによって、アカウントバケットのクォータを最大 1,000 バケットまで引き上げることができます。使用するバケットの数が多いか少ないかにかかわらず、パフォーマンスに違いはありません。

> **Note**
>
> それぞれの AWS リージョン に複数のクォータ引き上げリクエストを提出する必要はありません。バケットクォータが自分の AWS アカウント に適用されます。

バケットのクォータを引き上げる方法については、AWS 全般のリファレンスのの「AWS のサービスのクォータ」を参照してください。

バケット名の再利用

バケットが空の場合は削除できます。バケットが削除されると、その名前は再利用できるようになります。ただし、バケットを削除すると、さまざまな理由で名前を再利用できない場合があります。

例えば、バケットを削除してその名前が再利用可能になると、別の AWS アカウント がその名前でバケットを作成する場合があります。また、削除したバケットの名前が再利用可能になるまでに、しばらく時間がかかる場合もあります。同じバケット名を使用する場合は、バケットを削除しないことをお勧めします。

バケット名の詳細については、「バケットの名前付け」を参照してください。

オブジェクトとバケットの制限

最大バケットサイズまたはバケットに保存できるオブジェクト数に制限はありません。すべてのオブジェクトを 1 つのバケットに格納してもかまいませんし、複数のバケットに分けて整理してもかまいません。ただし、別のバケット内からバケットを作成することはできません。

バケットオペレーション

Amazon S3 の高可用性技術は、get、put、list、および delete オペレーションに重点を置いています。バケットのオペレーションは、一元化されグローバルなリソース空間に対して作用するため、アプリケーションの可用性の高いコードパスでは、バケットの作成、削除、設定を行うことは推奨されません。バケットの作成、削除、設定は、個別の初期化や、頻繁に実行しないセットアップルーチンで実行する方が適しています。

バケットの命名規則と自動作成されたバケット

アプリケーションが自動的にバットを作成する場合は、命名の衝突が起きないようなバケット命名スキームを使用してください。特定のバケット名が既に使用されている場合は、異なるバケット名をアプリケーションロジックが選択するようにしてください。

バケットの命名の詳細については、「バケットの名前付け」を参照してください。

Amazon S3 でのオブジェクトのアップロード、ダウンロード、操作

Amazon S3 にデータを保存するには、バケットおよびオブジェクトと呼ばれるリソースを使用します。バケット とは、オブジェクトのコンテナのことです。オブジェクト とは、ファイルと、そのファイルを記述している任意のメタデータのことです。

Amazon S3 にオブジェクトを保存するには、バケットを作成してから、オブジェクトをバケットにアップロードします。オブジェクトがバケットの中にあるときは、オブジェクトを開き、ダウンロードして、コピーできます。オブジェクトまたはバケットが不要になった場合は、これらのリソースをクリーンアップできます。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

> **Important**
>
> Amazon S3 コンソールで、オブジェクトに [Open] (開く) または [Download As] (名前を付けてダウンロード) を選択すると、これらのオペレーションによって署名付き URL が作成されます。5 分間、これらの署名付き URL にアクセスできる人なら誰でもオブジェクトにアクセスできるようになります。署名付き URL の詳細については、「署名付き URL の使用」を参照してください。

Amazon S3 では、お支払いは実際に使用した分のみです。Amazon S3 の機能と料金の詳細については、「Amazon S3」を参照してください。Amazon S3 の新規のお客様は、Amazon S3 を無料で使い始めることができます。詳細については、AWS 無料利用枠 を参照してください。

トピック

- Amazon S3 オブジェクトの概要
- オブジェクトキー名の作成
- オブジェクトメタデータの使用

- オブジェクトのアップロード
- マルチパートアップロードを使用したオブジェクトのアップロードとコピー
- オブジェクトのコピー、移動、名前の変更
- オブジェクトのダウンロード
- オブジェクトの整合性をチェックする
- Amazon S3 オブジェクトの削除
- オブジェクトの整理、リスト化、使用
- 署名付き URL の使用
- S3 Object Lambda を使用したオブジェクトの変換

Amazon S3 オブジェクトの概要

Amazon S3 は、一意のキー値を使用して、必要な数のオブジェクトを保存できるオブジェクトストアです。これらのオブジェクトは 1 つ以上のバケットに保存し、各オブジェクトのサイズは最大 5 TB です。オブジェクトの構成エレメントを以下に示します。

キー

オブジェクトに割り当てる名前です。オブジェクトを取得するには、オブジェクトキーを使用します。詳細については、Working with Object Metadata を参照してください。

バージョン ID

バケット内ではキーとバージョン ID がオブジェクトを一意に識別します。バージョン ID は、バケットにオブジェクトを追加すると Amazon S3 によって生成される文字列です。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

値

保存するコンテンツです。

オブジェクトの値は任意のバイトシーケンスです。オブジェクトのサイズは、0 から 5 TB までの範囲です。詳細については、「オブジェクトのアップロード」を参照してください。

メタデータ

オブジェクトに関する情報を保存するために使用できる名前と値のペアのセットです。Amazon S3 内のオブジェクトには、ユーザー定義メタデータと呼ばれるメタデータを割り当てることが

できます。Amazon S3 では、オブジェクトの管理目的で使用されるシステムメタデータもオブジェクトに割り当てられます。詳細については、Working with Object Metadata を参照してください。

サブリソース

Amazon S3 では、オブジェクト固有の追加情報を保存するためにサブリソースメカニズムを使用します。サブリソースはオブジェクトの従属物なので、オブジェクトやバケットといった他のエンティティに常に関連付けられます。詳細については、「オブジェクトのサブリソース」を参照してください。

アクセスコントロールの情報

Amazon S3 に保存するオブジェクトへのアクセスを制御できます。Amazon S3 は、アクセスコントロールリスト (ACL) やバケットポリシーなどのリソースベースのアクセスコントロールと、ユーザーベースのアクセスコントロールをサポートしています。アクセスコントロールの詳細については、以下を参照してください。

- アクセス管理
- Amazon S3 用 Identity and Access Management
- ACL の設定

Amazon S3 リソース (バケットやオブジェクトなど) はデフォルトでプライベートです。他のユーザーがそれらのリソースにアクセスできるようにするには、明示的にアクセス許可を与える必要があります。オブジェクトの共有の詳細については、署名付き URL を使用したオブジェクトの共有 を参照してください。

タグ

タグを使用して、保存されているオブジェクトを分類し、アクセスコントロールやコスト配分を行うことができます。詳細については、「タグを使用してストレージを分類する」を参照してください。

オブジェクトのサブリソース

Amazon S3 では、バケットとオブジェクトに関連付ける一連のサブリソースが定義されています。サブリソースはオブジェクトの下位に属します。つまり、サブリソースは単独では存在しません。常に、オブジェクトやバケットなどの他のエンティティに関連付けられています。

次の表は、Amazon S3 のオブジェクトに関連付けられるサブリソースをまとめたものです。

サブリソース	説明
acl	アクセス許可の被付与者と付与されるアクセス許可を識別するリストが含まれます。オブジェクトを作成するとき、acl を使用することで、オブジェクト所有者がオブジェクトに対して完全なコントロールを持つことを指定します。オブジェクト ACL の取得や、更新されたアクセス許可のリストで置き換えることができます。ACL を更新するときは必ず既存の ACL を置き換える必要があります。ACL の詳細については、アクセスコントロールリスト (ACL) の概要 を参照してください。

オブジェクトキー名の作成

オブジェクトキー (またはキー名) によって、Amazon S3 バケット内のオブジェクトは一意に識別されます。オブジェクトメタデータは、名前と値のペアのセットです。オブジェクトメタデータの詳細については、オブジェクトメタデータの使用 を参照してください。

オブジェクトを作成するときに、キー名を指定します。これにより、バケット内のオブジェクトは一意に識別されます。例えば、Amazon S3 コンソールでは、バケットを強調表示すると、そのバケットに含まれるオブジェクトのリストが表示されます。表示される名前がオブジェクトキーです。オブジェクトキー名は一連の Unicode 文字で、UTF-8 にエンコードすると最大で 1,024 バイト長になります。オブジェクトキー名では大文字と小文字が区別されます。

> **Note**
>
> 値「soap」を持つオブジェクトキー名は、仮想ホスト形式のリクエストについてサポートされていません。「soap」が使用されるオブジェクトキー名値の場合、パス形式の URL を代わりに使用する必要があります。

Amazon S3 のデータモデルはフラットな構造をしています。バケットを作成し、バケットにオブジェクトを保存します。サブバケットやサブフォルダの階層はありません。ただし、Amazon S3 コンソールで使用されているようなキー名のプレフィックスや区切り記号を使用して論理的な階層を暗示できます。Amazon S3 コンソールでは、フォルダの概念をサポートしています。Amazon S3 コンソールからメタデータを編集する方法の詳細については、Amazon S3 コンソールでのオブジェクトメタデータの編集 を参照してください。

バケット (admin-created) に、次のようなオブジェクトキーを持つ 4 つのオブジェクトがあるとします。

```
Development/Projects.xls

Finance/statement1.pdf

Private/taxdocument.pdf

s3-dg.pdf
```

コンソールでは、キー名のプレフィックス (Development/、Finance/、Private/) および区切り記号 ("/") を使用して、フォルダ構造を表します。s3-dg.pdf キーにはプレフィックスがないため、そのオブジェクトはバケットのルートレベルに直接表示されます。Development/ フォルダを開くと、Projects.xlsx オブジェクトが表示されます。

- Amazon S3 ではバケットとオブジェクトをサポートしており、階層はありません。ただし、オブジェクトキー名のプレフィックスと区切り記号により、Amazon S3 コンソールや AWS SDK で階層を暗示し、フォルダの概念を導入できます。
- Amazon S3 コンソールは、フォルダのプレフィックスと区切り記号の値をキーとしてゼロバイトのオブジェクトを作成することにより、フォルダオブジェクトの作成を実装します。これらのフォルダオブジェクトはコンソールに表示されません。それ以外の場合は、他のオブジェクトと同様に動作し、REST API、AWS CLI、および AWS SDK を使用して表示および操作できます。

オブジェクトキーの命名のガイドライン

オブジェクトキー名には UTF-8 文字を使用できます。ただし、キー名に特定の文字を使用すると、一部のアプリケーションやプロトコルで問題が発生することがあります。以下のガイドラインは、DNS、ウェブセーフ文字、XML パーサー、その他の API とのコンプライアンスを最大化するのに役立ちます。

セーフ文字

以下の文字セットは、一般的にキー名で使用しても安全です。

Alphanumeric characters	• 0-9 • a ~ z

	• A ~ Z
Special characters	• 感嘆符 (!) • ハイフン (-) • 下線 (_) • ピリオド (.) • アスタリスク (*) • 一重引用符 (') • 丸かっこ開き (() • 丸かっこ閉じ ())

有効なオブジェクトキー名の例を次に示します。

- `4my-organization`
- `my.great_photos-2014/jan/myvacation.jpg`
- `videos/2014/birthday/video1.wmv`

> **Note**
>
> Amazon S3 コンソールを使用してダウンロードしたキー名がピリオド「.」で終わるオブジェクトは、ダウンロードしたオブジェクトのキー名からピリオド「.」が削除されます。ダウンロードしたオブジェクトのキー名の末尾をピリオド「.」に保持したままダウンロードするには、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用します。
> また、以下のプレフィックスの制約についても注意してください。
>
> - プレフィックスが「./」のオブジェクトは、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用してアップロードまたはダウンロードします。Amazon S3 コンソールを使用することはできません。
> - プレフィックスが「../」のオブジェクトは、AWS Command Line Interface (AWS CLI) または Amazon S3 コンソールを使用してアップロードすることはできません。

特殊な処理を必要とする可能性がある文字

キー名で以下の文字を使用すると、追加のコード処理が必要になる場合があり、16 進数として URL エンコードまたは参照することが必要になる可能性があります。これらの文字の一部は表示不可能な文字であり、ブラウザで処理されない場合があります。この場合も、特殊な処理が必要です。

- アンパサンド ("&")
- ドル記号 ("$")
- 16 進数の 00 ~ 1F (10 進数の 0 ~ 31) の範囲および 7F (10 進数の 127) の ASCII 文字
- アットマーク ("@")
- 等号 ("=")
- セミコロン (";")
- スラッシュ (/)
- コロン (":")
- プラス記号 ("+")
- スペース – いくつかの用途 (特に複数のスペース) では、スペースの重要なシーケンスが失われる可能性があります。
- カンマ (",")
- 疑問符 ("?")

使用しない方がよい文字

すべてのアプリケーションで一貫性を維持するには相当な量の特殊な処理が必要になるため、キー名には以下の文字を使用しないことをお勧めします。

- バックスラッシュ ("\")
- 左中括弧 ("{")
- 表示不可能な ASCII 文字 (10 進数の 128 ~ 255 の文字)
- カレット ("^")
- 右中括弧 ("}")
- パーセント記号 ("%")
- アクサングラーブ/バックティック ("`")
- 右角括弧 ("]")

- 引用符
- 大なり記号 (">")
- 左角括弧 ("[")
- チルダ ("~")
- 小なり記号 ("<")
- シャープ記号 ("#")
- 縦棒/パイプ ("|")

XML 関連のオブジェクトキーの制約

行末処理に関する XML 標準で規定されているとおり、すべての XML テキストは正規化され、1 つのキャリッジリターン (ASCII コード 13) と改行の直後のキャリッジリターン (ASCII コード 10) が単一の改行文字に置き換えられます。XML リクエストでオブジェクトキーを正しく解析するには、キャリッジリターンやその他の特殊文字を XML タグ内に挿入するときに、同等の XML エンティティコードに置き換える必要があります。以下では、当該特殊文字とそれに相当するエンティティコードのリストを示しています。

- ' (')
- ” (")
- & (&)
- < (<)
- > (>)
- \r (
 または
)
- \n (
 または
)

Example

次の例では、キャリッジリターンの代わりに XML エンティティコードを使用する方法を示しています。この DeleteObjects リクエストにより、key パラメータ: /some/prefix/objectwith\rcarriagereturn (\r はキャリッジリターン) を持つオブジェクトが削除されます。

```
<Delete xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Object>
    <Key>/some/prefix/objectwith&#13;carriagereturn</Key>
  </Object>
```

```
</Delete>
```

オブジェクトメタデータの使用

Amazon S3 では、オブジェクトをアップロードする時にオブジェクトメタデータを設定できます。オブジェクトメタデータは、名前と値のペアのセットです。オブジェクトのアップロード後にはオブジェクトメタデータは変更できません。オブジェクトメタデータを変更する唯一の方法は、オブジェクトのコピーを作成し、メタデータを設定することです。

オブジェクトを作成するときに、キー名も指定します。これにより、バケット内のオブジェクトは一意に識別されます。オブジェクトキー (またはキー名) によって、Amazon S3 バケット内のオブジェクトは一意に識別されます。詳細については、「オブジェクトキー名の作成」を参照してください。

Amazon S3 には、システム定義メタデータとユーザー定義メタデータの 2 種類のメタデータがあります。以下のセクションでは、システム定義メタデータとユーザー定義のメタデータについて詳しく説明します。Amazon S3 コンソールを使用したメタデータの編集の詳細については、Amazon S3 コンソールでのオブジェクトメタデータの編集 を参照してください。

システムで定義されたオブジェクトメタデータ

Amazon S3 では、バケットに格納されるオブジェクトごとに、一連のシステムメタデータが維持されます。Amazon S3 はこのシステムメタデータを必要に応じて処理します。例えば、Amazon S3 ではオブジェクトの作成日とサイズに関するメタデータが維持され、オブジェクト管理の目的でこの情報が使用されます。

システムメタデータには 2 つのカテゴリがあります。

- システム制御 – オブジェクトの作成日などのメタデータはシステムによって制御され、Amazon S3 だけがその値を変更できることを意味します。
- ユーザー制御 – 他のシステムメタデータ (オブジェクトに設定済みのストレージクラスや、オブジェクトでサーバー側の暗号化が有効になっているかどうかなど) は、ユーザーが値を制御するシステムメタデータの例です。バケットをウェブサイトとして設定している場合、ページリクエストを別のページや外部 URL にリダイレクトしたいことがあります。この場合、ウェブページはバケット内のオブジェクトです。Amazon S3 は、ページリダイレクト値をシステムメタデータとして保存し、ユーザーがその値を制御します。

  オブジェクトを作成するときに、このようなシステムメタデータ項目の値を設定したり、必要に応じてその値を更新したりできます。ストレージクラスの詳細については、Amazon S3 ストレージクラスを使用する を参照してください。

Amazon S3 は Amazon S3 オブジェクトを暗号化する AWS KMS キーを使用します。AWS KMS はオブジェクトデータのみを暗号化します。チェックサムは、指定されたアルゴリズムとともに、オブジェクトのメタデータの一部として保存されます。サーバー側でオブジェクトに対する暗号化が要求された場合、チェックサムは暗号化された形式で保存されます。サーバーサイドの暗号化の詳細については、「暗号化によるデータの保護」を参照してください。

> **Note**
>
> PUT リクエストヘッダーのサイズは 8 KB に制限されています。PUT リクエストヘッダー内のシステム定義メタデータのサイズは 2 KB に制限されています。システム定義メタデータのサイズは、キーと値の各ペアを US-ASCII にエンコードしたバイト数の合計に基づいて測定されます。

次の表は、システム定義のメタデータのリストとユーザーがそれを更新できるかどうかをまとめたものです。

名前	説明	ユーザーが値を変更することはできますか?
`Date`	現在の日付と時刻。	なし
`Cache-Control`	キャッシュポリシーを指定するために使用される一般的なヘッダーフィールド。	あり
`Content-Disposition`	オブジェクトのプレゼンテーション情報。	あり
`Content-Length`	オブジェクトのサイズ (バイト単位)。	なし
`Content-Type`	オブジェクトのタイプ。	あり
`Last-Modified`	オブジェクト作成日または最終更新日のいずれか遅い方。マルチパートアップロードの場合、オブジェクトの作成日はマルチパートアップロードの開始日です。	なし

名前	説明	ユーザーが値を変更することはできますか?
ETag	オブジェクトのエンティティタグ (ETag) は、そのオブジェクトの特定のバージョンを表します。マルチパートアップロードとしてアップロードされず、暗号化されていない、または Simple Storage Service (Amazon S3) 管理キー (SSE-S3) を使用したサーバー側の暗号化によって暗号化されているオブジェクトの場合、ETag はデータの MD5 ダイジェストです。	なし
x-amz-server-side-encryption	オブジェクトでサーバー側の暗号化が有効であるか、そして暗号化が AWS Key Management Service (AWS KMS) または Amazon S3 で管理された暗号化 (SSE-S3) によるものかを示すヘッダー。詳細については、「サーバー側の暗号化によるデータの保護」を参照してください。	あり
x-amz-checksum-crc32 , x-amz-checksum-crc32c , x-amz-checksum-sha1 , x-amz-checksum-sha256	オブジェクトのチェックサムまたはダイジェストが含まれるヘッダー。Amazon S3 に使用するように指示したチェックサムアルゴリズムに応じて、これらのヘッダーの最大 1 つが一度に設定されます。チェックサムアルゴリズムの選択の詳細については、「オブジェクトの整合性をチェックする」を参照してください。	なし
x-amz-version-id	オブジェクトのバージョン。バケットのバージョニングを有効にすると、Amazon S3 ではバケットに追加されたオブジェクトにバージョン ID が割り当てられます。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。	なし

名前	説明	ユーザーが値を変更することはできますか?
x-amz-delete-marker	オブジェクトが削除マーカーであるかどうかを示すブール値マーカー。このマーカーは、バージョニングが有効になっているバケットでのみ使用されます。	なし
x-amz-storage-class	オブジェクトの保存に使われるストレージクラス。詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。	あり
x-amz-website-redirect-location	関連付けられたオブジェクトのリクエストを同じバケット内の別のオブジェクトまたは外部 URL にリダイレクトするヘッダー。詳細については、「(オプション) ウェブページリダイレクトの設定」を参照してください。	あり
x-amz-server-side-encryption-aws-kms-key-id	オブジェクトの暗号化に使用された AWS KMS 対称暗号化 KMS キーの ID を示すヘッダー。このヘッダーは、x-amz-server-side-encryption ヘッダーが存在し、値が aws:kms である場合にのみ使用されます。	あり
x-amz-server-side-encryption-customer-algorithm	お客様が用意した暗号化キーを使用したサーバー側の暗号化 (SSE-C) が有効であるかどうかを示すヘッダー。詳細については、「お客様が指定したキーによるサーバー側の暗号化 (SSE-C) の使用」を参照してください。	あり
x-amz-tagging	オブジェクトのタグセット。タグセットは URL クエリパラメータとしてエンコードする必要があります。	あり

ユーザー定義のオブジェクトメタデータ

オブジェクトをアップロードするときに、そのオブジェクトにメタデータを割り当てることもできます。このオプション情報は、オブジェクトを作成するための PUT リクエストまたは POST リクエストを送信するときに、名前と値 (キーと値) のペアとして指定します。REST API を使用してオブジェクトをアップロードするときは、オプションのユーザー定義メタデータ名を「x-amz-meta-」

で始め、他の HTTP ヘッダーと区別する必要があります。REST API を使用してオブジェクトを取得するときは、このプレフィックスが返されます。SOAP API を使用してオブジェクトをアップロードする場合、このプレフィックスは不要です。SOAP API を使用してオブジェクトを取得するときは、オブジェクトのアップロードに使用した API にかかわらず、プレフィックスは削除されます。

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

REST API を介してメタデータを取得するとき、同じ名前を持つヘッダー (大文字と小文字は区別されません) は Amazon S3 によってカンマ区切りリストに結合されます。出力不可能な文字が含まれているメタデータは返されません。その代わりに、`x-amz-missing-meta` ヘッダーが、出力不可能なメタデータエントリの数を示す値と共に返されます。`HeadObject` アクションは、オブジェクト自体を返さずに、オブジェクトからメタデータを取得します。このオペレーションは、オブジェクトのメタデータのみに関心がある場合に役立ちます。`HEAD` を使用するには、オブジェクトへの `READ` アクセス権が必要です。詳細については、Amazon Simple Storage Service API リファレンスの「HeadObject」を参照してください。

ユーザー定義メタデータはキーと値のペアのセットです。Amazon S3 はユーザー定義メタデータキーを小文字で保存します。

Amazon S3 では、任意の Unicode 文字をメタデータ値で使用できます。

これらのメタデータ値の表示に伴う問題を回避するために、REST を使用する場合は US-ASCII 文字を使用してください。SOAP またはブラウザベースのアップロードを `POST` 経由で使用する場合は UTF-8 を使用してください。

US-ASCII 以外の文字をメタデータ値で使用すると、指定した Unicode 文字列に US-ASCII 以外の文字が含まれていないかどうか調べられます。そのようなヘッダーの値は、保存される前に RFC 2047 に従って文字デコードされます。さらに、RFC 2047 に従ってエンコードされ、メールセーフになってから返されます。文字列に US-ASCII 文字のみが含まれている場合は、そのまま表示されます。

次に例を示します。

```
PUT /Key HTTP/1.1
```

```
Host: DOC-EXAMPLE-BUCKET1.s3.amazonaws.com
x-amz-meta-nonascii: ÄMÄZÕÑ S3

HEAD /Key HTTP/1.1
Host: DOC-EXAMPLE-BUCKET1.s3.amazonaws.com
x-amz-meta-nonascii: =?UTF-8?B?w4PChE3Dg8KEWsODwpXDg8KRIFMz?=

PUT /Key HTTP/1.1
Host: DOC-EXAMPLE-BUCKET1.s3.amazonaws.com
x-amz-meta-ascii: AMAZONS3

HEAD /Key HTTP/1.1
Host: DOC-EXAMPLE-BUCKET1.s3.amazonaws.com
x-amz-meta-ascii: AMAZONS3
```

> **Note**
>
> PUT リクエストヘッダーのサイズは 8 KB に制限されています。PUT リクエストヘッダー内のユーザー定義メタデータのサイズは 2 KB に制限されています。ユーザー定義メタデータのサイズは、キーと値それぞれの UTF-8 エンコーディングの合計バイト数を使用して測定されます。

アップロード済みオブジェクトのメタデータの変更 (オブジェクトのコピーを作成し、それを変更して古いオブジェクトと置き換える場合、または新しいバージョンを作成する場合) については、Amazon S3 コンソールでのオブジェクトメタデータの編集 を参照してください。

Amazon S3 コンソールでのオブジェクトメタデータの編集

Amazon S3 コンソールを使用して、既存の S3 オブジェクトのメタデータを編集できます。一部のメタデータは、オブジェクトをアップロードするときに Amazon S3 によって設定されます。例えば、`Content-Length` および `Last-Modified` は、ユーザーが変更できない、システム定義のオブジェクトメタデータフィールドです。

また、オブジェクトのアップロード時にメタデータを設定し、必要に応じて後で編集することもできます。例えば、STANDARD ストレージクラスに最初に保存したオブジェクトのセットがあるとします。時間の経過とともに、このデータを高可用性にする必要がなくなる可能性があります。そのため、`x-amz-storage-class` キーの値を STANDARD から GLACIER に編集して、ストレージクラスを GLACIER に変更します。

Note

Amazon S3 でオブジェクトメタデータを編集する場合は、次の問題を考慮してください。

- このアクションにより、更新された設定と最終更新日を持つオブジェクトのコピーが作成されます。S3 バージョニングが有効になっている場合は、オブジェクトの新しいバージョンが作成され、既存のオブジェクトが古いバージョンになります。S3 バージョニングが有効になっていない場合は、元のオブジェクトがオブジェクトの新しいコピーに置き換えられます。プロパティを変更する IAM ロールに関連付けられた AWS アカウント も、新しいオブジェクトまたは (オブジェクトバージョン) の所有者になります。
- Amazon S3 コンソールを使用してユーザー定義タグを持つオブジェクトのメタデータを編集するには、`s3:GetObjectTagging` のアクセス許可も必要です。Amazon S3 コンソールを使用して、ユーザー定義タグがなく、サイズが 16 MB を超えるオブジェクトのメタデータを編集する場合は、`s3:GetObjectTagging` のアクセス許可も必要です。

  ターゲットバケットポリシーによって `s3:GetObjectTagging` アクションが拒否されると、オブジェクトのメタデータは更新されますが、ユーザー定義タグはオブジェクトから削除され、エラーが発生します。
- メタデータを編集すると、既存のキー名の値が更新されます。
- お客様が用意した暗号化キー (SSE-C) で暗号化されたオブジェクトは、コンソールを使用してコピーできません。AWS CLI、AWS SDK、または Amazon S3 REST API を使用する必要があります。

Warning

フォルダのメタデータを編集する場合は、`Edit metadata` オペレーションが完了するのを待ってから、新しいオブジェクトをフォルダに追加します。そうしないと、新しいオブジェクトも編集される可能性があります。

以下のトピックでは、Amazon S3 コンソールを使用してオブジェクトのメタデータを編集する方法について説明します。

システム定義メタデータの編集

S3 オブジェクトのシステムメタデータは、一部のみ設定できます。システム定義のメタデータのリストと値を変更できるかどうかは、システムで定義されたオブジェクトメタデータ を参照してください。

オブジェクトのシステム定義メタデータを編集するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. Amazon S3 バケットまたはフォルダに移動し、編集するメタデータを持つオブジェクト名の左側にあるチェックボックスをオンにします。
3. [アクション] メニューで [アクションの編集] を選択し、[メタデータの編集] を選択します。
4. リストされたオブジェクトを確認し、[メタデータの追加] を選択します。
5. メタデータの [Type (タイプ)] には、[System-defined (システム定義)] を選択します。
6. 一意の [Key (キー)] とメタデータの [Value (値)] を指定します。
7. 追加のメタデータを編集するには、[Add metadata (メタデータの追加)] を選択します。[削除] をクリックして、タイプ、キー、値のセットを削除することもできます。
8. 完了したら、[Edit metadata] (メタデータの編集) を選択すると、指定したオブジェクトのメタデータを Amazon S3 が編集します。

ユーザー定義メタデータの編集

メタデータプレフィックス、`x-amz-meta-`、およびカスタムキーを作成するために選択した名前を組み合わせて、オブジェクトのユーザー定義メタデータを編集できます。例えば、カスタム名 `alt-name` を追加した場合、メタデータキーは `x-amz-meta-alt-name` になります。

ユーザー定義メタデータの合計サイズは最大 2 KB です。ユーザー定義メタデータの合計サイズを計算するには、キーと値ごとの UTF-8 エンコーディングのバイト数を合計します。キーと値の両方が US-ASCII 標準に従っている必要があります。詳細については、「ユーザー定義のオブジェクトメタデータ」を参照してください。

オブジェクトのユーザー定義メタデータを編集するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

2. [バケット] リストで、メタデータを追加するオブジェクトが含まれるバケットの名前を選択します。

   必要に応じてフォルダに移動することもできます。
3. [オブジェクト] リストで、メタデータを追加するオブジェクト名の横にあるチェックボックスをオンにします。
4. [アクション] メニューで、[メタデータの編集] を選択します。
5. リストされたオブジェクトを確認し、[メタデータの追加] を選択します。
6. メタデータの [タイプ] には [ユーザー定義] を選択します。
7. `x-amz-meta-` に続いて、一意のカスタム [キー] を入力します。メタデータの [Value (値)] も入力します。
8. さらにメタデータを追加するには、[Add metadata (メタデータの追加)] を選択します。[削除] をクリックして、タイプ、キー、値のセットを削除することもできます。
9. [Edit metadata (メタデータの編集)] を選択します。

   Amazon S3 は、指定されたオブジェクトのメタデータを編集します。

オブジェクトのアップロード

Amazon S3 にファイルをアップロードすると、S3 オブジェクトとして保存されます。オブジェクトは、オブジェクトを記述するファイルデータとメタデータから構成されます。バケット内のオブジェクトの数に制限はありません。Amazon S3 バケットにファイルをアップロードするには、バケットに対する書き込みアクセス許可が必要です。アクセス許可の詳細については、Amazon S3 用 Identity and Access Management を参照してください。

ファイルタイプ (イメージ、バックアップ、データ、ムービーなど) を問わず、各種のファイルを S3 バケットにアップロードできます。Amazon S3 コンソールを使用すると、アップロードできるファイルの最大サイズが 160 GB になります。160 GB を超えるファイルをアップロードするには、AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 REST API を使用します。

バージョニングが有効なバケットに既に存在するキー名の付いたオブジェクトをアップロードした場合、Amazon S3 は既存のオブジェクトを置き換える代わりにオブジェクトの別バージョンを作成します。バージョニングの詳細については、「S3 コンソールの使用」を参照してください。

アップロードするデータのサイズに応じて、Amazon S3 には以下のオプションが用意されています。

- AWS SDK、REST API、または AWS CLI を使用して 1 回のオペレーションでオブジェクトをアップロードする — 1 回の PUT オペレーションでは、最大 5 GB の単一のオブジェクトをアップロードできます。
- Amazon S3 コンソールを使用して 1 つのオブジェクトをアップロードする — Amazon S3 コンソールでは、最大 160 GB のオブジェクトをアップロードできます。
- AWS SDK、REST API、または AWS CLI を使用してオブジェクトをいくつかに分けてアップロードする — マルチパートアップロード API を使用すると、最大 5 TB のサイズの単一の大容量オブジェクトをアップロードできます。

  マルチパートアップロード API オペレーションは大容量オブジェクトのアップロードを効率よく行えるように設計されています。1 つのオブジェクトをいくつかに分けてアップロードできます。オブジェクトのパートは、単独で、任意の順序で、または並行してアップロードできます。マルチパートアップロードは 5 MB ~ 5 TB のオブジェクトで使用できます。詳細については、「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。

オブジェクトをアップロードすると、そのオブジェクトは、デフォルトでは Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) を使用して自動的に暗号化されます。ダウンロードすると、オブジェクトは復号化されます。詳細については、Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定および暗号化によるデータの保護を参照してください。

オブジェクトをアップロードするときに、別の種類のデフォルト暗号化を使用する場合は、S3 PUT リクエストで AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS) を指定するか、SSE-KMS を使用してデータを暗号化するように送信先バケットのデフォルトの暗号化設定を設定することもできます。SSE-KMS に関する詳細は、「AWS KMS (SSE-KMS) によるサーバー側の暗号化の指定」を参照してください。別のアカウントが所有している KMS キーを使用する場合は、そのキーを使用するアクセス許可が必要です。KMS キーのクロスアカウント権限の詳細については、「AWS Key Management Service デベロッパーガイド」の「他のアカウントで使用できる KMS キーを作成する」を参照してください。

Amazon S3 でアクセス拒否 (403 Forbidden) エラーが発生したとき、一般的な原因の詳細については「Amazon S3 でのアクセス拒否 (403 Forbidden) エラーのトラブルシューティング 」を参照してください。

S3 コンソールの使用

この手順では、コンソールを使用してオブジェクトおよびフォルダを Amazon S3 バケットにアップロードする方法について説明します。

オブジェクトをアップロードする場合、オブジェクトキー名はファイル名および任意のプレフィックスになります。Amazon S3 コンソールでは、フォルダを作成してオブジェクトを整理できます。Amazon S3 では、フォルダはオブジェクトキー名に表示されるプレフィックスとして表されます。Amazon S3 コンソールのフォルダに個々のオブジェクトをアップロードする場合、フォルダ名はオブジェクトキー名に含まれます。

例えば、「sample1.jpg」という名前のオブジェクトを「backup」という名前のフォルダにアップロードすると、キー名は「backup/sample1.jpg」になります。ただし、オブジェクトはコンソールの sample1.jpg フォルダ内で backup として表示されます。有効なキー名の詳細については、オブジェクトメタデータの使用 を参照してください。

> **Note**
>
> Amazon S3 コンソールでオブジェクトの名前を変更したり、ストレージクラス、暗号化、またはメタデータなどのプロパティを変更したりすると、新しいオブジェクトが作成され、古いオブジェクトが置き換えられます。S3 バージョニングが有効になっている場合は、オブジェクトの新しいバージョンが作成され、既存のオブジェクトが古いバージョンになります。また、プロパティを変更するロールは、新しいオブジェクト (またはオブジェクトのバージョン) の所有者になります。

フォルダをアップロードすると、Amazon S3 は、そのフォルダからすべてのファイルとサブフォルダをバケットにアップロードします。その後、アップロードしたファイルの名前とフォルダの名前を組み合わせたオブジェクトキー名が割り当てられます。例えば、/images と sample1.jpg の 2 つのファイルを含む sample2.jpg というフォルダをアップロードすると、Amazon S3 はファイルのアップロード後に、対応するキー名である images/sample1.jpg と images/sample2.jpg を割り当てます。キー名にはプレフィックスとしてフォルダ名が含まれています。Amazon S3 コンソールには、最後の / に続くキー名の部分のみが表示されます。例えば、images フォルダ内では images/sample1.jpg オブジェクトと images/sample2.jpg オブジェクトが sample1.jpg および sample2.jpg として表示されます。

フォルダとファイルを S3 バケットにアップロードするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [Buckets (バケット)] リストで、フォルダやファイルのアップロード先のバケットの名前を選択します。

4. [アップロード] を選択します。
5. [Upload] (アップロード) ウィンドウで、次のいずれかの操作を行います。
   - ファイルとフォルダを [Upload] (アップロード) ウィンドウにドラッグアンドドロップします。
   - [ファイルの追加] または [フォルダの追加] を選択し、アップロードするファイルまたはフォルダを選択して [開く] を選択します。
6. バージョニングを有効にするには、[Destination] (送信先) で [Enable Bucket Versioning] (バケットバージョニングを有効化) を選択します。
7. 追加のアップロードオプションを設定せずにリストされたファイルとフォルダをアップロードするには、ページの下部で [Upload] (アップロード) を選択します。

   Amazon S3 はオブジェクトとフォルダをアップロードします。アップロードが完了すると、[アップロード: ステータス] ページに成功のメッセージが表示されます。

追加のオブジェクトプロパティを設定するには

1. アクセスコントロールリストの許可を変更するには、[Permissions] (許可) を選択します。
2. [Access control list (ACL)] (アクセスコントロールリスト (ACL) で、許可を編集します。

   オブジェクトのアクセス許可については、S3 コンソールを使用した、オブジェクトの ACL アクセス権限の設定 を参照してください。アップロードするすべてのファイルについて、オブジェクトの読み取りアクセスをパブリック (世界中のすべてのユーザー) に付与できます。ただし、パブリック読み取りアクセスのデフォルト設定を変更しないことをお勧めします。パブリック読み取りアクセス権限の付与は、バケットがウェブサイトなどに使用されるなど、ユースケースの小さいサブセットに適用されます。オブジェクトをアップロードした後で、オブジェクトの許可をいつでも変更できます。
3. その他の追加プロパティを設定するには、[Properties] (プロパティ) を選択します。
4. [ストレージクラス] で、アップロードするファイルのストレージクラスを選択します。

   ストレージクラスの詳細については、Amazon S3 ストレージクラスを使用する を参照してください。
5. オブジェクトの暗号化設定を更新するには、[Server-side encryption settings] (サーバー側の暗号化設定) で、次の操作を行います。

   a. [Specify an encryption key] (暗号化キーを指定する) を選択します。

b. [暗号化設定] で、[デフォルトの暗号化にバケット設定を使用する] または [デフォルトの暗号化にバケット設定を上書きする] を選択します。

c. [デフォルトの暗号化にバケット設定を上書きする] を選択した場合は、次の暗号化設定を設定する必要があります。

- Amazon S3 管理のキーを使用してアップロードされたファイルを暗号化するには、[Amazon S3 マネージドキー (SSE-S3)] を選択します。

  詳細については、「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)」を参照してください。

- AWS Key Management Service (AWS KMS) に保存されているキーを使用してアップロード済みファイルを暗号化するには、AWS Key Management Service キー (SSE-KMS) を選択します。次に、AWS KMS キーについて、以下のいずれかのオプションを選択します。

  - 使用可能な KMS キーのリストから選択するには、[AWS KMS keys から選択する] を選択し、使用可能なキーのリストから自分の KMS キーを選択します。

    AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がこのリストに表示されます。カスタマーマネージドキーの詳細については、AWS Key Management Serviceデベロッパーガイドの「カスタマーキーと AWS キー」を参照してください。

  - KMS キー ARN を入力するには、[AWS KMS key ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。

  - AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

    AWS KMS key の作成の詳細については、AWS Key Management Service デベロッパーガイドのキーの作成を参照してください。

> **Important**
>
> バケットと同じ AWS リージョン で使用可能な KMS キーのみを使用できます。Amazon S3 コンソールには、バケットと同じリージョンで最初の 100 個の KMS キーしか表示されません。リストに存在しない KMS キーを使用するには、KMS キー ARN を入力する必要があります。別のアカウントが所有している KMS キーを使用する場合は、まずそのキーを使用するアクセス許可が必要であり、次に KMS キー ARN を入力する必要があります。

Amazon S3 では、対称暗号化 KMS キーのみがサポートされ、非対称暗号化 KMS キーはサポートされません。詳細については、AWS Key Management Service デベロッパーガイドの「Identifying symmetric and asymmetric KMS keys」(対称および非対称 KMS キーの識別) を参照してください。

6. 追加のチェックサムを使用するには、[On] (オン) を選択します。次に、[Checksum function] (チェックサム関数) で、使用する関数を選択します。Amazon S3 は、オブジェクト全体を受け取った後、チェックサム値を計算して保存します。[Precalculated value] (事前計算された値) ボックスを使用して、事前計算された値を指定できます。その場合、Amazon S3 は、指定した値と計算した値を比較します。2 つの値が一致しない場合、Amazon S3 はエラーを生成します。

   追加のチェックサムを使用すると、データの検証に使用するチェックサムアルゴリズムを指定できます。追加のチェックサムの詳細については、「オブジェクトの整合性をチェックする」を参照してください。

7. アップロードするすべてのオブジェクトにタグを追加するには、[Add tag (タグの追加)] を選択します。[キー] フィールドにタグ名を入力します。タグの値を入力します。

   オブジェクトのタグ付けにより、ストレージを分類する方法が提供されます。各タグはキーバリューのペアです。キーとタグ値は大文字と小文字が区別されます。オブジェクトごとに最大 10 個のタグを持つことができます。タグキーには最大 128 個の Unicode 文字、タグ値には最大 255 個の Unicode 文字を使用できます。オブジェクトタグの詳細については、タグを使用してストレージを分類するを参照してください。

8. メタデータを追加するには、[Add metadata] (メタデータの追加) を選択します。

   a. [Type] (タイプ) で、[System defined] (システム定義) または [User defined] (ユーザー定義) を選択します。

      システム定義のメタデータの場合は、Content-Type や Content-Disposition などの一般的な HTTP ヘッダーを選択できます。システム定義のメタデータのリストと値を追加できるかどうかについては、システムで定義されたオブジェクトメタデータ を参照してください。プレフィックス `x-amz-meta-` で始まるメタデータはすべてユーザー定義のメタデータとして扱われます。ユーザー定義メタデータはオブジェクトと共に保存され、オブジェクトのダウンロード時に返されます。キーと値の両方が US-ASCII 標準に従っている必要があります。ユーザー定義メタデータのサイズは最大 2 KB です。システム定義メタデータとユーザー定義メタデータの詳細については、オブジェクトメタデータの使用 を参照してください。

b. [Key] (キー) で、キーを選択します。

c. キーの値を入力します。

9. オブジェクトをアップロードするには、[Upload] (アップロード) を選択します。

   Amazon S3 はオブジェクトをアップロードします。アップロードが完了すると、[Upload: status] (アップロード: ステータス) ページに成功のメッセージが表示されます。

10. [終了] を選択します。

AWS SDK の使用

AWS SDK を使用して、Amazon S3 にオブジェクトをアップロードできます。SDK にはデータを容易にアップロードできるラッパーライブラリが用意されています。詳細については、サポートされている SDK のリスト を参照してください。

次に、いくつかの SDK を選択した例を数例挙げます。

.NET

次の C# コード例では、2 つの `PutObjectRequest` リクエストで 2 つのオブジェクトを作成します。

- 最初の `PutObjectRequest` リクエストでは、サンプルオブジェクトデータとしてテキスト文字列を保存します。また、バケット名とオブジェクトキー名を指定します。
- 2 番目の `PutObjectRequest` リクエストでは、ファイル名を指定してファイルをアップロードします。また、このリクエストは、`ContentType` ヘッダーとオプションのオブジェクトメタデータ (タイトル) を指定します。

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadObjectTest
```

```
   {
        private const string bucketName = "*** bucket name ***";
        // For simplicity the example creates two objects from the same file.
        // You specify key names for these objects.
        private const string keyName1 = "*** key name for first object created ***";
        private const string keyName2 = "*** key name for second object created
***";
        private const string filePath = @"*** file path ***";
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.EUWest1;

        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            WritingAnObjectAsync().Wait();
        }

        static async Task WritingAnObjectAsync()
        {
            try
            {
                // 1. Put object-specify only key name for the new object.
                var putRequest1 = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName1,
                    ContentBody = "sample text"
                };

                PutObjectResponse response1 = await
client.PutObjectAsync(putRequest1);

                // 2. Put the object-set ContentType and add metadata.
                var putRequest2 = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName2,
                    FilePath = filePath,
                    ContentType = "text/plain"
                };

                putRequest2.Metadata.Add("x-amz-meta-title", "someTitle");
```

```
                PutObjectResponse response2 = await
 client.PutObjectAsync(putRequest2);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an
 object"
                        , e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Unknown encountered on server. Message:'{0}' when writing an
 object"
                    , e.Message);
            }
        }
    }
}
```

Java

次の例では 2 つのオブジェクトを作成します。最初のオブジェクトにはデータとしてテキスト文字列があり、2 番目のオブジェクトはファイルです。この例では、AmazonS3Client.putObject() への呼び出しでバケット名、オブジェクトキー、およびテキストデータを直接指定して、最初のオブジェクトを作成します。例では、バケット名、オブジェクトキー、およびファイルパスを指定する PutObjectRequest を指定して 2 番目のオブジェクトを作成します。PutObjectRequest では、ContentType ヘッダーとタイトルメタデータも指定します。

作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;
```

```
import java.io.File;
import java.io.IOException;

public class UploadObject {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String stringObjKeyName = "*** String object key name ***";
        String fileObjKeyName = "*** File object key name ***";
        String fileName = "*** Path to file to upload ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .build();

            // Upload a text string as a new object.
            s3Client.putObject(bucketName, stringObjKeyName, "Uploaded String
 Object");

            // Upload a file as a new object with ContentType and title specified.
            PutObjectRequest request = new PutObjectRequest(bucketName,
 fileObjKeyName, new File(fileName));
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentType("plain/text");
            metadata.addUserMetadata("title", "someTitle");
            request.setMetadata(metadata);
            s3Client.putObject(request);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

JavaScript

次の例では、特定のリージョンの Amazon S3 バケットに既存のファイルをアップロードします。

```
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new PutObjectCommand({
    Bucket: "test-bucket",
    Key: "hello-s3.txt",
    Body: "Hello S3!",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

PHP

この例では、AWS SDK for PHP のクラスを使用して、5 GB までのサイズのオブジェクトをアップロードする手順を示します。ファイルのサイズが大きい場合には、マルチパートアップロード API オペレーションを使用する必要があります。詳細については、「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。

AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

Example - データをアップロードして Amazon S3 バケットにオブジェクトを作成する

以下の PHP コード例では、`putObject()` メソッドを使用してデータをアップロードすることで、指定されたバケットにオブジェクトを作成しています。

```
require 'vendor/autoload.php';
```

```
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

try {
    // Upload data.
    $result = $s3->putObject([
        'Bucket' => $bucket,
        'Key'    => $keyname,
        'Body'   => 'Hello, world!',
        'ACL'    => 'public-read'
    ]);

    // Print the URL to the object.
    echo $result['ObjectURL'] . PHP_EOL;
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

AWS SDK for Ruby – バージョン 3 には、Amazon S3 にオブジェクトをアップロードする 2 つの方法があります。1 つ目の方法では、ディスクから任意のサイズのファイルを簡単にアップロードできる、マネージド型ファイルアップローダーを使用します。マネージド型ファイルアップローダーによる方法を使用するには、次の操作を行います。

1. `Aws::S3::Resource` クラスのインスタンスを作成します。
2. バケット名とキーで、ターゲットオブジェクトを参照します。オブジェクトはバケット内に保持され、各オブジェクトを特定するための一意のキーを持っています。
3. オブジェクトで `#upload_file` を呼び出します。

Example

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

AWS SDK for Ruby – バージョン 3 でオブジェクトをアップロードできる 2 つ目の方法では、#put の Aws::S3::Object メソッドを使用します。この方法は、オブジェクトが文字列であるか、ディスク上のファイルではない I/O オブジェクトである場合に役立ちます。この方法を使用するには、次の操作を行います。

1. Aws::S3::Resource クラスのインスタンスを作成します。
2. バケット名とキーで、ターゲットオブジェクトを参照します。
3. #put を呼び出し、文字列または I/O オブジェクトを渡します。

Example

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
```

```
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

REST API の使用

REST リクエストを送信してオブジェクトをアップロードできます。PUT リクエストを送信して 1 回のオペレーションでデータをアップロードできます。詳細については、「PutObject」を参照してください。

AWS CLI の使用

単一のオペレーションで、最大 5 GB のオブジェクトをアップロードする PUT リクエストを送信できます。詳細については、`PutObject`AWS CLI コマンドリファレンスでの例を参照してください。

マルチパートアップロードを使用したオブジェクトのアップロードとコピー

マルチパートアップロードを使用すると、単一のオブジェクトをパートのセットとしてアップロードすることができます。各パートは、オブジェクトのデータの連続する部分です。これらのオブジェクトパートは、任意の順序で個別にアップロードできます。いずれかのパートの送信が失敗すると、他のパートに影響を与えることなくそのパートを再送することができます。オブジェクトのすべてのパートがアップロードされたら、Amazon S3 はこれらのパートを組み立ててオブジェクトを作成します。通常、オブジェクトサイズが 100 MB 以上の場合は、単一のオペレーションでオブジェクトをアップロードする代わりに、マルチパートアップロードを使用することを考慮してください。

マルチパートアップロードの使用には、次の利点があります。

- スループットの向上 – パートを並列にアップロードすることで、スループットを向上させることができます。
- ネットワーク問題からの迅速な回復 – パートサイズが小さいほど、ネットワークエラーにより失敗したアップロードを再開する際の影響を最小限に抑えることができます。

- オブジェクトのアップロードの一時停止と再開 – オブジェクトの複数のパートを徐々にアップロードできます。マルチパートアップロードを開始した後は終了期限がありません。マルチパートアップロードは明示的に完了または停止する必要があります。
- オブジェクトの最終的なサイズが不明な状態でアップロードを開始 – オブジェクトの作成中でもアップロードを開始できます。

次の方法でマルチパートアップロードを使用することをお勧めします。

- 安定した高帯域幅ネットワーク経由で大きなオブジェクトをアップロードする場合は、複数スレッドのパフォーマンスのために並行してオブジェクトパートをアップロードすることにより、マルチパートアップロードを使用して使用可能な帯域幅の使用を最大化します。
- むらがあるネットワークでアップロードを実行する場合は、マルチパートアップロードを使用して、アップロードの再開を回避することで、ネットワークエラーに対する弾力性を高めます。マルチパートアップロードを使用するときには、アップロード中に中断されたパートのアップロードを再試行するだけで済みます。最初からオブジェクトのアップロードを再開する必要はありません。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。S3 Express One Zone とディレクトリバケットでマルチパートアップロードを使用する方法の詳細については、「ディレクトリバケットでのマルチパートアップロードの使用」を参照してください。

マルチパートアップロードのプロセス

マルチパートアップロードは 3 つのステップで構成されるプロセスです。まずアップロードを開始し、次にオブジェクトのパートをアップロードします。すべてのパートをアップロードしたらマルチパートアップロードを完了します。Amazon S3 の側では、マルチパートアップロードの完了リクエストを受け取ると同時に、アップロードされたパートからオブジェクトを構築します。構築されたオブジェクトは、バケット内の他のオブジェクトと同じようにアクセスできます。

進行中のすべてのマルチパートアップロードをリストしたり、特定のマルチパートアップロードにおいてアップロードが完了したパートのリストを取得したりできます。このようなオペレーションのそれぞれについて、このセクションで説明します。

マルチパートアップロードの開始

リクエストを送信すると、アップロード ID を含むレスポンスが Amazon S3 から返されます。アップロード ID はマルチパートアップロードの一意の識別子です。パートのアップロード、パートのリスト、アップロードの完了、アップロードの停止を行うときは常に、このアップロード ID を指定する必要があります。アップロードするオブジェクトの説明となるメタデータを指定する場合は、マルチパートアップロードの開始リクエストの中にそれを指定する必要があります。

パートのアップロード

パートをアップロードするときは、アップロード ID に加えて、パート番号を指定する必要があります。1 ~ 10,000 の範囲で任意のパート番号を選択できます。パート番号によって、アップロードするオブジェクトに含まれるパートとその位置が一意に識別されます。選択するパート番号は、連続している必要はありません (例えば、1、5、14 など)。以前にアップロードしたパートと同じパート番号を使って新しいパートをアップロードした場合、以前のパートは上書きされます。

パートをアップロードしたときに、Amazon S3 はエンティティタグ (ETag) ヘッダーを含むレスポンスを返します。パートのアップロードごとに、パート番号と ETag 値を記録する必要があります。マルチパートアップロードを完了するためには、残りのリクエストにこれらの値を含める必要があります。各パートには、アップロード時に独自の ETag が設定されます。ただし、マルチパートアップロードが完了し、すべてのパートが統合されると、チェックサムのチェックサムとしてすべてのパートが 1 つの ETag の下に置かれます。

> **Note**
>
> マルチパートアップロードを開始して、1 つまたは複数のパートをアップロードした後は、マルチパートアップロードを完了するか停止しない限り、アップロードしたパートのストレージに対する課金を停止できません。マルチパートアップロードを完了または停止した後でのみ、Amazon S3 はパートのストレージを解放して、パートのストレージに対する課金を停止します。
>
> マルチパートアップロードを停止した後は、再度同じアップロード ID を使ってパートをアップロードすることはできません。パートのアップロードが進行しているときにアップロードを停止した後も、パートのアップロードは成功または失敗する可能性があります。すべてのパートによって使用されているストレージをすべて解放するには、すべてのパートのアップロードが完了した後でマルチパートアップロードを停止する必要があります。

マルチパートアップロードの完了

マルチパートアップロードを完了すると、パート番号に基づいて昇順に連結されたオブジェクトが Amazon S3 によって作成されます。マルチパートアップロードの開始リクエストにオブジェクトメタデータが指定されている場合、Amazon S3 によってそのメタデータはオブジェクトに関連付けられます。完了リクエストが正常に処理されると、個々のパートはなくなります。

マルチパートアップロードの完了リクエストには、アップロード ID と、パート番号およびそれに対応する ETag 値の両方のリストが含まれている必要があります。Amazon S3 からのレスポンスには、結合されるオブジェクトデータを一意に識別する ETag が含まれます。この ETag が、オブジェクトデータの MD5 ハッシュになるとは限りません。

マルチパートアップロードのサンプル呼び出し

この例では、100 GB ファイルのマルチパートアップロードを生成していると仮定します。この場合、プロセス全体に対して次の API コールが実行されます。合計 1002 回の API コールが実行されます。

- プロセスを開始する `CreateMultipartUpload` の呼び出し。
- それぞれ 100 MB の一部をアップロードし、合計サイズは 100 GB となる、1000 回の個別の `UploadPart` の呼び出し。
- プロセスを終了する `CompleteMultipartUpload` の呼び出し。

マルチパートアップロードのリスト化

特定のマルチパートアップロードのパートや、進行中のすべてのマルチパートアップロードをリスト表示できます。パートのリストオペレーションでは、特定のマルチパートアップロードについて既にアップロードしたパートの情報が返されます。パートのリストリクエストを送信するたびに、指定したマルチパートアップロードのパート情報 (最大で 1,000 個のパート) が Amazon S3 から返されます。マルチパートアップロードに 1,000 個を超えるパートが含まれる場合、すべてのパートを取得するにはパートのリストリクエストを追加で送信する必要があります。返されるパートのリストには、アップロードが完了していないパートは含まれていないことに注意してください。マルチパートアップロードの一覧表示オペレーションを使用すると、進行中のマルチパートアップロードのリストを取得できます。

進行中のマルチパートアップロードとは、開始されているものの、まだ完了または停止されていないアップロードを意味します。各リクエストに最大 1,000 個のマルチパートアップロードが返されます。進行中のマルチパートアップロードが 1,000 個を超える場合、残りのマルチパートアップロードを取得するには、リクエストを追加で送信する必要があります。返されたリストは確認の目的でのみ使用してください。マルチパートアップロードの完了リクエストを送信するときに、このリストの

結果を使用しないでください。代わりに、パートのアップロード時に指定したパート番号と、それに対応する、Amazon S3 から返される ETag 値の独自のリストを維持しておいてください。

マルチパートアップロードオペレーションを使用したチェックサム

Amazon S3 にオブジェクトをアップロードするときに、使用する Amazon S3 のチェックサムアルゴリズムを指定できます。Amazon S3 はデフォルトで MD5 を使用してデータの整合性を検証しますが、使用する追加のチェックサムアルゴリズムを指定することができます。MD5 を使用する場合、Amazon S3 はアップロードの完了後にマルチパートオブジェクト全体のチェックサムを計算します。このチェックサムはオブジェクト全体のチェックサムではなく、個々のパートのチェックサムのチェックサムです。

追加のチェックサムを使用するように Amazon S3 に指示すると、Amazon S3 は各パートのチェックサム値を計算し、その値を保存します。API または SDK を使用し、`GetObject` または `HeadObject` を使用して、個々のパートのチェックサム値を取得できます。まだ処理中のマルチパートアップロードの個々の部分のチェックサム値を取得したい場合は、`ListParts` を使用できます。

> **Important**
>
> 追加のチェックサムを含むマルチパートアップロードを使用している場合は、マルチパート番号には連続するパート番号を使用する必要があります。追加のチェックサムを使用する場合、連続しないパート番号でマルチパートアップロードリクエストを完了しようとすると、Amazon S3 は HTTP `500 Internal Server Error` エラーを生成します。

マルチパートオブジェクトでのチェックサムの動作の詳細については、「オブジェクトの整合性をチェックする」を参照してください。

マルチパートアップロードの同時オペレーション

分散開発環境においては、アプリケーションから同じオブジェクトに対して複数の更新が同時に開始されることもありえます。同じオブジェクトキーを使ってアプリケーションから複数のマルチパートアップロードが開始される可能性もあります。そのようなアップロードごとに、アプリケーションからパートのアップロードが行われ、アップロードの完了リクエストが Amazon S3 に送信されて、オブジェクトが作成されます。バケットで S3 バージョニングが有効になっているときには、マルチパートアップロードを完了するたびに新しいバージョンが作成されます。バージョニングが有効になっていないバケットの場合は、マルチパートアップロードの開始から完了までの間に受信された他の何らかのリクエストが優先される可能性もあります。

> **Note**
>
> マルチパートアップロードの開始から完了までの間に受信された他の何らかのリクエストが優先されることもありえます。例えば、あるキーを使ってマルチパートアップロードを開始した後、アップロードが完了しないうちに別のオペレーションによってそのキーが削除されたとします。その場合、オブジェクトを確認できなくても、マルチパートアップロードの完了レスポンスによってオブジェクト作成の成功が示される可能性があります。

マルチパートアップロードと料金

マルチパートアップロードを開始すると、アップロードを完了または中止するまですべてのパートが Amazon S3 によって保持されます。マルチパートアップロードの実行期間を通して、アップロードとそれに関連するパートのために使用されるすべてのストレージ、帯域幅、リクエストに対して課金が行われます。

これらのパートは、パートのアップロード時に指定されたストレージクラスに従って課金されます。ただし、S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive にアップロードされたパートは例外です。S3 Glacier Flexible Retrieval ストレージクラスへの PUT の処理中のマルチパートパートは、アップロードが完了するまで S3 Glacier Flexible Retrieval ステージングストレージとして S3 標準ストレージ料金で請求されます。さらに、CreateMultipartUpload と UploadPart は、どちらも S3 標準料金で請求されます。CompleteMultibleUpload リクエストのみが、S3 Glacier Flexible Retrieval 料金で請求されます。同様に、S3 Glacier Deep Archive ストレージクラスへの PUT の処理中のマルチパートパートは、アップロードが完了するまで S3 Glacier Flexible Retrieval ステージングストレージとして S3 標準ストレージ料金で請求されます。CompleteMultipartUpload リクエストのみが、S3 Glacier Deep Archive 料金で請求されます。

マルチパートアップロードを停止した場合、アップロードアーティファクトおよびアップロードしたすべてのパートは Amazon S3 によって削除され、それらに対して課金されることはなくなります。指定されたストレージクラスに関係なく、不完全なマルチパートアップロードの削除に伴う早期削除料金はありません。料金に関する詳細については、[Amazon S3 の料金] を参照してください。

> **Note**
>
> ストレージコストを最小限に抑えるため、`AbortIncompleteMultipartUpload` アクションを使用して指定した日数が経過した後に不完全なマルチパートアップロードを削除するようにライフサイクルルールを設定することをお勧めします。不完全なマルチパートアップロードを削除するライフサイクルルールの作成の詳細については、「不完全なマルチパート

アップロードを中止するためのバケットライフサイクルポリシーの設定」を参照してください。

マルチパートアップロードの API サポート

これらのライブラリは、マルチパートオブジェクトのアップロードを容易にする、高レベルの抽象化を実現します。ただし、アプリケーションの必要に応じて REST API を直接使用することもできます。Amazon Simple Storage Service API リファレンスの以下のセクションでは、マルチパートアップロードの REST API について説明しています。

AWS Lambda 関数を使用するマルチパートアップロードのチュートリアルについては、「Uploading large objects to Amazon S3 using multipart upload and transfer acceleration」を参照してください。

- マルチパートアップロードを作成する
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- マルチパートアップロードの中止
- パートのリスト
- マルチパートアップロードのリスト

マルチパートアップロードの AWS Command Line Interface サポート

マルチパートアップロードのオペレーションについては、AWS Command Line Interface の以下のトピックを参照してください。

- Initiate Multipart Upload
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- マルチパートアップロードの中止
- パートのリスト
- マルチパートアップロードのリスト

マルチパートアップロードの AWS SDK サポート

AWS SDK を使用して、オブジェクトを部分的にアップロードできます。API アクションでサポートされる AWS SDK のリストについては、次を参照してください。

- マルチパートアップロードを作成する
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- マルチパートアップロードの中止
- パートのリスト
- マルチパートアップロードのリスト

マルチパートアップロード API とアクセス許可

マルチパートアップロードオペレーションを使用するには、必要なアクセス権限を有している必要があります。マルチパートアップロードオペレーションを実行するためのアクセス許可を付与するには、アクセスコントロールリスト (ACL)、バケットポリシー、ユーザーポリシーを使用できます。ACL、バケットポリシー、またはユーザーポリシーを使用して割り当てることのできる、さまざまなマルチパートアップロードオペレーションに必要なアクセス許可を次の表に示します。

アクション	必要なアクセス許可
マルチパートアップロードの作成	マルチパートアップロードを作成するには、オブジェクトに対して `s3:PutObject` アクションを実行するための許可が必要です。 バケット所有者は他のプリンシパルに対して `s3:PutObject` アクションの実行を許可できます。
Initiate Multipart Upload	マルチパートアップロードを開始するには、オブジェクトに対して `s3:PutObject` アクションを実行するための許可が必要です。 バケット所有者は他のプリンシパルに対して `s3:PutObject` アクションの実行を許可できます。

アクション	必要なアクセス許可
イニシエータ	マルチパートアップロード開始者を識別するコンテナエレメント。イニシエータが AWS アカウント である場合、このエレメントは所有者エレメントと同じ情報を提供します。イニシエータが IAM ユーザーである場合、このエレメントはユーザー ARN と表示名を提供します。
Upload Part	パートをアップロードするには、オブジェクトに対して s3:PutObject アクションを実行するための許可が必要です。 マルチパートアップロードの開始者がオブジェクトのパートをアップロードできるようにするため、バケット所有者はその開始者に対しオブジェクトへの s3:PutObject アクションの実行を許可する必要があります。
パートのアップロード (コピー)	パートをアップロードするには、オブジェクトに対して s3:PutObject アクションを実行するための許可が必要です。既存のオブジェクトからパートをアップロードするので、ソースオブジェクトに対して s3:GetObject を実行するための許可が必要です。 開始者がオブジェクトのパートをアップロードするには、バケット所有者が、開始者にそのオブジェクトでの s3:PutObject アクションの実行を許可する必要があります。
Complete Multipart Upload	マルチパートアップロードを完了するには、オブジェクトに対して s3:PutObject アクションを実行するための許可が必要です。 マルチパートアップロードの開始者がオブジェクトのアップロードを完了できるようにするため、バケット所有者はその開始者に対しオブジェクトへの s3:PutObject アクションの実行を許可する必要があります。

アクション	必要なアクセス許可
マルチパートアップロードの中止	マルチパートアップロードを停止するには、s3:AbortMultipartUpload アクションを実行するための許可が必要です。 デフォルトでは、バケット所有者とマルチパートアップロードの開始者が、IAM とバケットポリシーの一部として、このアクションの実行を許可されます。開始者が IAM ユーザーである場合、そのユーザーの AWS アカウント もそのマルチパートアップロードを停止できます。VPC エンドポイントポリシーでは、マルチパートアップロードのイニシエータは、s3:AbortMultipartUpload アクションを実行する許可を自動的に取得しません。 このようなデフォルト設定に加え、バケット所有者は他のプリンシパルに対してオブジェクトへの s3:AbortMultipartUpload アクションの実行を許可できます。バケット所有者は任意のプリンシパルに対し、s3:AbortMultipartUpload アクションを実行する権限を無効にすることができます。
パートのリスト	マルチパートアップロードに含まれるパートをリストするには、s3:ListMultipartUploadParts アクションを実行するための許可が必要です。 デフォルトではバケット所有者が、バケットに対する任意のマルチパートアップロードについてパートのリストを許可されています。マルチパートアップロードの開始者は、特定のマルチパートアップロードについてパートのリストを許可されます。マルチパートアップロードの開始者が IAM ユーザーである場合、その IAM ユーザーを管理している AWS アカウント もそのアップロードのパートのリストへのアクセス許可を付与されます。 このようなデフォルト設定に加え、バケット所有者は他のプリンシパルに対してオブジェクトへの s3:ListMultipartUploadParts アクションの実行を許可できます。バケット所有者は任意のプリンシパルに対し、s3:ListMultipartUploadParts アクションを実行する権限を無効にすることもできます。

アクション	必要なアクセス許可
マルチパートアップロードのリスト	バケットに対して進行中のマルチパートアップロードをリストするには、そのバケットに対して s3:ListBucketMultipartUploads アクションを実行するための許可が必要です。 デフォルト設定に加え、バケット所有者は他のプリンシパルに対してバケットへの s3:ListBucketMultipartUploads アクションの実行を許可できます。
AWS KMS 暗号化および復号関連のアクセス許可	AWS Key Management Service (AWS KMS) MS キーを使用して暗号化を伴うマルチパートアップロードを実行するには、キーの kms:Decrypt および kms:GenerateDataKey アクションに対する許可がリクエスタに必要です。マルチパートアップロードを完了する前に、暗号化されたファイル部分からデータを復号して読み取る必要があるため、Amazon S3 にはこれらの許可が必要です。 詳細については、AWS ナレッジセンターのAWS KMS key を使用した暗号化で Amazon S3 に大容量ファイルをアップロードするを参照してください。 IAM ユーザーまたはロールが KMS キーと同じ AWS アカウント にある場合、キーポリシーでこれらの許可が必要です。IAM ユーザーまたはロールが KMS キーとは異なるアカウントに属している場合、キーポリシーと IAM ユーザーまたはロールの両方に対する許可が必要です。

ACL アクセス権限とアクセスポリシーのアクセス許可との関係については、ACL アクセス許可とアクセスポリシーのアクセス許可のマッピング を参照してください。IAM ユーザー、グループ、ロール、ベストプラクティスの詳細については、IAM ユーザーガイドの「IAM ID (ユーザー、グループ、ロール)」を参照してください。

トピック

- 不完全なマルチパートアップロードを削除するためのバケットライフサイクル設定の設定
- マルチパートアップロードを使用したオブジェクトのアップロード
- 高レベルの .NET TransferUtility クラスを使用してディレクトリをアップロードする
- マルチパートアップロードのリスト化
- マルチパートアップロードの追跡

- マルチパートアップロードの中止
- マルチパートアップロードを使用したオブジェクトのコピー
- Amazon S3 マルチパートアップロードの制限

不完全なマルチパートアップロードを削除するためのバケットライフサイクル設定の設定

ベストプラクティスとして、ストレージコストを最小限に抑えるため、AbortIncompleteMultipartUpload アクションを使用してライフサイクルルールを設定することをお勧めします。マルチパートアップロードを中止する方法の詳細については、マルチパートアップロードの中止 を参照してください。

Amazon S3 は、バケットライフサイクルルールをサポートしています。このルールを使用すると、開始後指定された日数内に完了しないマルチパートアップロードを停止するよう Amazon S3 に指示できます。マルチパートアップロードが指定された時間内に完了しない場合、中止操作の対象となります。Amazon S3 は、マルチパートアップロードを中止し、マルチパートアップロードに関連付けられているすべてのパートを削除します。このルールは、既存のマルチパートアップロードと後で作成するマルチパートアップロードの両方に適用されます。

AbortIncompleteMultipartUpload アクションにルールを指定するライフサイクル設定の例を次に示します。

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Prefix></Prefix>
        <Status>Enabled</Status>
        <AbortIncompleteMultipartUpload>
          <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

この例のルールでは、Prefix エレメントの値 (オブジェクトキー名のプレフィックス) を指定しません。したがって、これはマルチパートアップロードを開始したバケット内のすべてのオブジェクトに適用されます。開始されてから 7 日以内に完了しなかったマルチパートアップロード、中止オペレーションのターゲットとなります。中止アクションは、完了したマルチパートアップロードに影響を与えません。

バケットライフサイクル設定についての詳細は、ストレージのライフサイクルの管理 を参照してください。

> **Note**
>
> マルチパートアップロードが、ルールで指定された日数内に完了した場合、`AbortIncompleteMultipartUpload` ライフサイクルアクションは適用されません (つまり Amazon S3 はアクションを実行しません)。また、このアクションはオブジェクトには適用されません。このライフサイクルアクションによって削除されるオブジェクトはありません。また、不完全なマルチパートアップロード部分を削除しても、S3 ライフサイクルの早期削除料金は発生しません。

S3 コンソールの使用

未完了のマルチパートアップロードを自動的に管理するため、S3 コンソールを使用して、指定された日数後にバケットから未完了のマルチパートアップロードのバイトを失効させるライフサイクルポリシーを作成できます。次の手順は、7 日後に未完了のマルチパートアップロードを削除するためのライフサイクルルールを追加する方法を示しています。ライフサイクルルールの追加に関する詳細については、「バケットにライフサイクル設定を設定する」を参照してください。

7 日以上経過した未完了のマルチパートアップロードを中止するライフサイクルルールを追加するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets (バケット)] リストで、ライフサイクルルールを作成するバケットの名前を選択します。
3. [Management (管理)] タブを選択して、[Create lifecycle rule (ライフサイクルルールを作成する)] を選択します。
4. [Lifecycle rule name (ライフサイクルルール名)] に、ルールの名前を入力します。

   その名前はバケット内で一意である必要があります。
5. ライフサイクルルールのスコープを選択します。

   - 特定のプレフィックスを持つすべてのオブジェクトに対してライフサイクルルールを作成するには、[1 つまたは複数のフィルタを使用してこのルールの適用範囲を制限] を選択し、[プレフィックス] にプレフィックスを入力します。

- このライフサイクルルールをバケット内のすべてのオブジェクトに作成するには、[This rule applies to all objects in the bucket] (このルールをバケット内のすべてのオブジェクトに適用する) を選択して、[I acknowledge that this lifecycle rule will apply to all objects in the bucket] (このライフサイクルルールがバケット内のすべてのオブジェクトに適用されることを了承します) を選択します。

6. [Lifecycle rule actions] (ライフサイクルルールアクション) で、[Delete expired object delete markers or incomplete multipart uploads] (期限切れのオブジェクト削除マーカーまたは未完了のマルチパートアップロードを削除) を選択します。
7. [Delete expired object delete markers or incomplete multipart uploads] (期限切れオブジェクト削除マーカーまたは未完了のマルチパートアップロードを削除する) で、[Delete incomplete multipart uploads] (未完了のマルチパートアップロードを削除) を選択します。
8. [Number of days] (日数) フィールドに、未完了のマルチパートアップロードを削除するまでの日数 (この例では 7 日) を入力します。
9. ルールの作成を選択します。

AWS CLI の使用

以下の `put-bucket-lifecycle-configuration` AWS Command Line Interface (AWS CLI) コマンドは、指定したバケットのライフサイクル設定を追加します。このコマンドを使用するには、*`user input placeholders`* を自分の情報に置き換えます。

```
aws s3api put-bucket-lifecycle-configuration  \
        --bucket example-s3-bucket1  \
        --lifecycle-configuration filename-containing-lifecycle-configuration
```

次の例は、AWS CLI を使用して未完了のマルチパートアップロードを中止するライフサイクルルールを追加する方法を示しています。これには、7 日以上経過した未完了のマルチパートアップロードを中止する JSON ライフサイクル設定の例が含まれています。

この例の CLI コマンドを使用するには、*`user input placeholders`* を自分の情報に置き換えてください。

未完了のマルチパートアップロードを中止するためのライフサイクルルールを追加するには

1. AWS CLI をセットアップします。手順については、AWS CLI を使用した Amazon S3 での開発を参照してください。

2. 次のサンプルライフサイクル設定をファイル (例、*lifecycle.json*) に保存します。このサンプル設定では、空のプレフィックスが指定されているため、バケット内のすべてのオブジェクトに適用されます。設定がオブジェクトのサブセットに制限されるようにプレフィックスを指定できます。

```
{
    "Rules": [
        {
            "ID": "Test Rule",
            "Status": "Enabled",
            "Filter": {
                "Prefix": ""
            },
            "AbortIncompleteMultipartUpload": {
                "DaysAfterInitiation": 7
            }
        }
    ]
}
```

3. 次の CLI コマンドを実行して、バケットにこのライフサイクル設定を指定します。

```
aws s3api put-bucket-lifecycle-configuration   \
--bucket example-s3-bucket1  \
--lifecycle-configuration file://lifecycle.json
```

4. ライフサイクル設定がバケットに設定されていることを確認するには、次の `get-bucket-lifecycle` コマンドを使用してライフサイクル設定を取得します。

```
aws s3api get-bucket-lifecycle  \
--bucket example-s3-bucket1
```

5. ライフサイクル設定を削除するには、次の `delete-bucket-lifecycle` コマンドを使用します。

```
aws s3api delete-bucket-lifecycle \
--bucket example-s3-bucket1
```

マルチパートアップロードを使用したオブジェクトのアップロード

マルチパートアップロードを使用して、プログラムで 1 つのオブジェクトを Amazon S3 にアップロードできます。

詳細については、次のセクションを参照してください。

AWS SDK (高レベル API) の使用

一部の AWS SDK は、マルチパートアップロードを完了するために必要なさまざまな API オペレーションを 1 つのオペレーションに結合することで、マルチパートアップロードを簡素化する高レベル API を公開しています。詳細については、「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。

マルチパートアップロードを一時停止して再開する場合、アップロード中にパートサイズを変更する場合、またはデータのサイズが事前にわからない場合は、低レベル API メソッドを使用します。マルチパートアップロード用の低レベル API メソッドは、追加の機能も提供します。詳細については、「AWS SDK の使用 (低レベル API)」を参照してください。

Java

大きなファイルをアップロードするには、`TransferManager` クラスを使用します。この高レベル API オペレーションは、ファイルまたはストリームからデータをアップロードできます。マルチパートアップロードで使用するパートのサイズ、パートをアップロードする際に使用する同時スレッドの数など、詳細なオプションも設定できます。省略可能なオブジェクトのプロパティ、ストレージクラス、またはアクセスコントロールリスト (ACL) を設定することもできます。これらの詳細なオプションを設定するには、`PutObjectRequest` および `TransferManagerConfiguration` クラスを使用します。

可能な場合、`TransferManager` は複数のスレッドを使用して、1 回のアップロードに含まれる複数のパートを 1 度にアップロードしようとします。これにより、大きなコンテンツサイズと高帯域を処理する場合にスループットが大幅に改善されます。

ファイルのアップロード機能に加えて、`TransferManager` クラスでは進行中のマルチパートアップロードを中止することができます。アップロードを開始した後は、そのアップロードを完了または中止するまで進行中とみなされます。`TransferManager` は、指定したバケットで指定した日時より前に開始された進行中のすべてのマルチパートアップロードを中止します。

> **Note**
>
> データのソースにストリームを使っている場合、TransferManager クラスは同時アップロードを実行しません。

次の例では、高レベルマルチパートアップロード Java API (TransferManager クラス) を使用して、オブジェクトをロードします。作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import java.io.File;

public class HighLevelMultipartUpload {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Object key ***";
        String filePath = "*** Path for file to upload ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
            TransferManager tm = TransferManagerBuilder.standard()
                    .withS3Client(s3Client)
                    .build();

            // TransferManager processes all transfers asynchronously,
            // so this call returns immediately.
```

```
            Upload upload = tm.upload(bucketName, keyName, new File(filePath));
            System.out.println("Object upload started");

            // Optionally, wait for the upload to finish before continuing.
            upload.waitForCompletion();
            System.out.println("Object upload complete");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

S3 バケットにファイルをアップロードするには、TransferUtility クラスを使用します。ファイルからデータをアップロードする場合は、オブジェクトのキー名を指定する必要があります。指定しないと、API ではキー名としてファイル名を使用します。ストリームからデータをアップロードする場合は、オブジェクトのキー名を指定する必要があります。

高度なアップロードオプション (パートのサイズ、複数のパーツを同時にアップロードする際のスレッド数、メタデータ、ストレージクラス、ACL など) を設定するには、TransferUtilityUploadRequest クラスを使用します。

> **Note**
>
> データのソースにストリームを使っている場合、TransferUtility クラスは同時アップロードを実行しません。

次の C# の例では、ファイルを複数のパートに分割して Amazon S3 バケットにアップロードします。さまざまな TransferUtility.Upload オーバーロードを使用してファイルをアップロードする方法を示します。後続のアップロード呼び出しが行われるたびに、前のアップロードが置き換えられます。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadFileMPUHighLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string keyName = "*** provide a name for the uploaded object
 ***";
        private const string filePath = "*** provide the full path name of the file
 to upload ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            UploadFileAsync().Wait();
        }

        private static async Task UploadFileAsync()
        {
            try
            {
                var fileTransferUtility =
                    new TransferUtility(s3Client);

                // Option 1. Upload a file. The file name is used as the object key
 name.
                await fileTransferUtility.UploadAsync(filePath, bucketName);
                Console.WriteLine("Upload 1 completed");

                // Option 2. Specify object key name explicitly.
                await fileTransferUtility.UploadAsync(filePath, bucketName,
 keyName);
                Console.WriteLine("Upload 2 completed");
```

```
                // Option 3. Upload data from a type of System.IO.Stream.
                using (var fileToUpload =
                    new FileStream(filePath, FileMode.Open, FileAccess.Read))
                {
                    await fileTransferUtility.UploadAsync(fileToUpload,
                                               bucketName, keyName);
                }
                Console.WriteLine("Upload 3 completed");

                // Option 4. Specify advanced settings.
                var fileTransferUtilityRequest = new TransferUtilityUploadRequest
                {
                    BucketName = bucketName,
                    FilePath = filePath,
                    StorageClass = S3StorageClass.StandardInfrequentAccess,
                    PartSize = 6291456, // 6 MB.
                    Key = keyName,
                    CannedACL = S3CannedACL.PublicRead
                };
                fileTransferUtilityRequest.Metadata.Add("param1", "Value1");
                fileTransferUtilityRequest.Metadata.Add("param2", "Value2");

                await fileTransferUtility.UploadAsync(fileTransferUtilityRequest);
                Console.WriteLine("Upload 4 completed");
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }

        }
    }
}
```

JavaScript

Example

サイズの大きいファイルをアップロードします。

```
import {
  CreateMultipartUploadCommand,
  UploadPartCommand,
  CompleteMultipartUploadCommand,
  AbortMultipartUploadCommand,
  S3Client,
} from "@aws-sdk/client-s3";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

export const main = async () => {
  const s3Client = new S3Client({});
  const bucketName = "test-bucket";
  const key = "multipart.txt";
  const str = createString();
  const buffer = Buffer.from(str, "utf8");

  let uploadId;

  try {
    const multipartUpload = await s3Client.send(
      new CreateMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );

    uploadId = multipartUpload.UploadId;

    const uploadPromises = [];
    // Multipart uploads require a minimum size of 5 MB per part.
    const partSize = Math.ceil(buffer.length / 5);

    // Upload each part.
```

```
    for (let i = 0; i < 5; i++) {
      const start = i * partSize;
      const end = start + partSize;
      uploadPromises.push(
        s3Client
          .send(
            new UploadPartCommand({
              Bucket: bucketName,
              Key: key,
              UploadId: uploadId,
              Body: buffer.subarray(start, end),
              PartNumber: i + 1,
            }),
          )
          .then((d) => {
            console.log("Part", i + 1, "uploaded");
            return d;
          }),
      );
    }

    const uploadResults = await Promise.all(uploadPromises);

    return await s3Client.send(
      new CompleteMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
        MultipartUpload: {
          Parts: uploadResults.map(({ ETag }, i) => ({
            ETag,
            PartNumber: i + 1,
          })),
        },
      }),
    );

    // Verify the output by downloading the file from the Amazon Simple Storage
Service (Amazon S3) console.
    // Because the output is a 25 MB string, text editors might struggle to open the
file.
  } catch (err) {
    console.error(err);
```

```
    if (uploadId) {
      const abortCommand = new AbortMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
      });

      await s3Client.send(abortCommand);
    }
  }
};
```

Example

サイズの大きいファイルをダウンロードします。

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream } from "fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: parseInt(start),
    end: parseInt(end),
    length: parseInt(length),
  };
};
```

```
export const isComplete = ({ end, length }) => end === length - 1;

// When downloading a large file, you might want to break it down into
// smaller pieces. Amazon S3 accepts a Range header to specify the start
// and end of the byte range to be downloaded.
const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    console.log(`Downloading bytes ${nextRange.start} to ${nextRange.end}`);

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
};

export const main = async () => {
  await downloadInChunks({
    bucket: "my-cool-bucket",
    key: "my-cool-object.txt",
  });
};
```

Go

Example

アップロードマネージャーを使用してデータを分割し、同時にアップロードすることで、大きなオブジェクトをアップロードすることができます。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}
```

```
// UploadLargeObject uses an upload manager to upload data to an object in a bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(bucketName string, objectKey string,
 largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }

 return err
}
```

Example

ダウンロードマネージャーを使用して、データを分割して取得し、同時にダウンロードすることで、大きなオブジェクトをダウンロードすることができます。

```
// DownloadLargeObject uses a download manager to download an object from a bucket.
```

```
// The download manager gets the data in parts and writes them to a buffer until all
 of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(bucketName string, objectKey string)
 ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader) {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(context.TODO(), buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

PHP

このトピックでは、マルチパートファイルのアップロードの Aws\S3\Model\MultipartUpload\UploadBuilder からの高レベルの AWS SDK for PHP クラスを使用する方法について説明します。AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

以下の PHP サンプルは、ファイルを Amazon S3 バケットにアップロードします。この例では、MultipartUploader オブジェクトのパラメータを設定する方法を示します。

```
 require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
```

```
    'region'  => 'us-east-1'
]);

// Prepare the upload parameters.
$uploader = new MultipartUploader($s3, '/path/to/large/file.zip', [
    'bucket' => $bucket,
    'key'    => $keyname
]);

// Perform the upload.
try {
    $result = $uploader->upload();
    echo "Upload complete: {$result['ObjectURL']}" . PHP_EOL;
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Python

次の例では、高レベルマルチパートアップロード Python API (`TransferManager` クラス) を使用して、オブジェクトをロードします。

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig


MB = 1024 * 1024
s3 = boto3.resource("s3")


class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """
```

```
    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

```
def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

```
def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
```

```
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
```

```
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

AWS SDK の使用 (低レベル API)

AWS SDK は、Amazon S3 REST API に非常によく似たマルチパートアップロード用の下位 API を公開しています (マルチパートアップロードを使用したオブジェクトのアップロードとコピー を参照)。マルチパートアップロードを一時停止して再開する必要がある場合やアップロード中にパートサイズを変更する場合、またはアップロードデータのサイズが事前にわからない場合は、低レベル API を使用します。これらの要件がない場合は、高レベル API (AWS SDK (高レベル API) の使用 を参照) を使用します。

Java

次の例では、低レベル Java クラスを使用してファイルをアップロードする方法を示します。以下のステップが実行されます。

- `AmazonS3Client.initiateMultipartUpload()` メソッドを使用してマルチパートアップロードを開始し、`InitiateMultipartUploadRequest` オブジェクトを渡します。
- `AmazonS3Client.initiateMultipartUpload()` メソッドが返すアップロード ID を保存します。以降、マルチパートアップロードオペレーションのたびに、このアップロード ID を指定する必要があります。
- オブジェクトのパートをアップロードします。パートごとに、`AmazonS3Client.uploadPart()` メソッドを呼び出します。`UploadPartRequest` オブジェクトを使用して、パートアップロード情報を指定します。
- 各パートで、リストの `AmazonS3Client.uploadPart()` メソッドのレスポンスから ETag を保存します。ETag 値を使用して、マルチパートアップロードを完了します。
- `AmazonS3Client.completeMultipartUpload()` メソッドを呼び出して、マルチパートアップロードを完了します。

Example

作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class LowLevelMultipartUpload {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String filePath = "*** Path to file to upload ***";

        File file = new File(filePath);
        long contentLength = file.length();
        long partSize = 5 * 1024 * 1024; // Set part size to 5 MB.

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Create a list of ETag objects. You retrieve ETags for each object
 part
            // uploaded,
            // then, after each individual part has been uploaded, pass the list of
 ETags to
            // the request to complete the upload.
            List<PartETag> partETags = new ArrayList<PartETag>();
```

```
            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
InitiateMultipartUploadRequest(bucketName, keyName);
            InitiateMultipartUploadResult initResponse =
s3Client.initiateMultipartUpload(initRequest);

            // Upload the file parts.
            long filePosition = 0;
            for (int i = 1; filePosition < contentLength; i++) {
                // Because the last part could be less than 5 MB, adjust the part
size as
                // needed.
                partSize = Math.min(partSize, (contentLength - filePosition));

                // Create the request to upload a part.
                UploadPartRequest uploadRequest = new UploadPartRequest()
                        .withBucketName(bucketName)
                        .withKey(keyName)
                        .withUploadId(initResponse.getUploadId())
                        .withPartNumber(i)
                        .withFileOffset(filePosition)
                        .withFile(file)
                        .withPartSize(partSize);

                // Upload the part and add the response's ETag to our list.
                UploadPartResult uploadResult = s3Client.uploadPart(uploadRequest);
                partETags.add(uploadResult.getPartETag());

                filePosition += partSize;
            }

            // Complete the multipart upload.
            CompleteMultipartUploadRequest compRequest = new
CompleteMultipartUploadRequest(bucketName, keyName,
                    initResponse.getUploadId(), partETags);
            s3Client.completeMultipartUpload(compRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }
}
```

.NET

次の C# の例では、低レベルの AWS SDK for .NET マルチパートアップロード API を使用して S3 バケットにファイルをアップロードする方法を示します。Amazon S3 のマルチパートアップロードの詳細については、マルチパートアップロードを使用したオブジェクトのアップロードとコピー を参照してください。

> **Note**
>
> AWS SDK for .NET API を使用して大きなオブジェクトをアップロードする場合、リクエストストリームへのデータの書き込み中に、タイムアウトが発生することがあります。`UploadPartRequest` を使用して、明示的なタイムアウトを設定できます。

次 C# の例では、低レベルのマルチパートアップロード API を使用して S3 バケットにファイルをアップロードします。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadFileMPULowLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string keyName = "*** provide a name for the uploaded object
 ***";
        private const string filePath = "*** provide the full path name of the file
 to upload ***";
```

```
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Uploading an object");
            UploadObjectAsync().Wait();
        }

        private static async Task UploadObjectAsync()
        {
            // Create list to store upload part responses.
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

            // Setup information required to initiate the multipart upload.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = bucketName,
                Key = keyName
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // Upload parts.
            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                Console.WriteLine("Uploading parts");

                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                        {
                            BucketName = bucketName,
```

```
                            Key = keyName,
                            UploadId = initResponse.UploadId,
                            PartNumber = i,
                            PartSize = partSize,
                            FilePosition = filePosition,
                            FilePath = filePath
                        };

                    // Track upload progress.
                    uploadRequest.StreamTransferProgress +=
                        new
EventHandler<StreamTransferProgressArgs>(UploadPartProgressEventCallback);

                    // Upload a part and add the response to our list.
                    uploadResponses.Add(await
s3Client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                // Setup to complete the upload.
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                    {
                        BucketName = bucketName,
                        Key = keyName,
                        UploadId = initResponse.UploadId
                     };
                completeRequest.AddPartETags(uploadResponses);

                // Complete the upload.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown: { 0}",
exception.Message);

                // Abort the upload.
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
                {
                    BucketName = bucketName,
```

```
                    Key = keyName,
                    UploadId = initResponse.UploadId
                };
                await s3Client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
        public static void UploadPartProgressEventCallback(object sender,
 StreamTransferProgressArgs e)
        {
            // Process event.
            Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
        }
    }
}
```

PHP

このトピックガイドでは、uploadPart のバージョン 3 から低レベルの AWS SDK for PHP メソッドを使用してファイルを複数のパートに分割してアップロードする方法について説明します。AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

次の PHP の例では、下位の PHP API マルチパートアップロードを使用して、Amazon S3 バケットにファイルをアップロードします。

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$filename = '*** Path to and Name of the File to Upload ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

$result = $s3->createMultipartUpload([
    'Bucket'       => $bucket,
    'Key'          => $keyname,
    'StorageClass' => 'REDUCED_REDUNDANCY',
```

```
    'Metadata'     => [
        'param1' => 'value 1',
        'param2' => 'value 2',
        'param3' => 'value 3'
    ]
]);
$uploadId = $result['UploadId'];

// Upload the file in parts.
try {
    $file = fopen($filename, 'r');
    $partNumber = 1;
    while (!feof($file)) {
        $result = $s3->uploadPart([
            'Bucket'     => $bucket,
            'Key'        => $keyname,
            'UploadId'   => $uploadId,
            'PartNumber' => $partNumber,
            'Body'       => fread($file, 5 * 1024 * 1024),
        ]);
        $parts['Parts'][$partNumber] = [
            'PartNumber' => $partNumber,
            'ETag' => $result['ETag'],
        ];
        $partNumber++;

        echo "Uploading part $partNumber of $filename." . PHP_EOL;
    }
    fclose($file);
} catch (S3Exception $e) {
    $result = $s3->abortMultipartUpload([
        'Bucket'   => $bucket,
        'Key'      => $keyname,
        'UploadId' => $uploadId
    ]);

    echo "Upload of $filename failed." . PHP_EOL;
}

// Complete the multipart upload.
$result = $s3->completeMultipartUpload([
    'Bucket'   => $bucket,
    'Key'      => $keyname,
    'UploadId' => $uploadId,
```

```
    'MultipartUpload'    => $parts,
]);
$url = $result['Location'];

echo "Uploaded $filename to $url." . PHP_EOL;
```

AWS SDK for Ruby の使用

AWS SDK for Ruby バージョン 3 では、2 つの方法で Amazon S3 マルチパートアップロードがサポートされています。1 つ目のオプションとして、マネージドファイルアップロードを使用できます。詳細については、AWS 開発者ブログの Amazon S3 にファイルをアップロードするを参照してください。マネージドファイルアップロードは、バケットにファイルをアップロードするための推奨される方法です。これらのアップロードには次の利点があります。

- 15 MB より大きなオブジェクトのマルチパートアップロードを管理します。
- エンコードの問題を回避するために、バイナリモードでファイルを正しく開きます。
- 大きなオブジェクトの複数パートを並行してアップロードするために、複数のスレッドを使用します。

次に示すマルチパートアップロードクライアントオペレーションを直接使用することもできます。

- create_multipart_upload – マルチパートアップロードを開始し、アップロード ID を返します。
- upload_part – マルチパートアップロードでパートをアップロードします。
- upload_part_copy – データソースとして既存のオブジェクトからデータをコピーすることで、パートをアップロードします。
- complete_multipart_upload – 以前にアップロードしたパートを組み合わせることで、マルチパートアップロードを完了します。
- abort_multipart_upload – マルチパートアップロードを停止します。

REST API の使用

Amazon Simple Storage Service API リファレンスの以下のセクションでは、マルチパートアップロードの REST API について説明しています。

- Initiate Multipart Upload
- Upload Part

- Complete Multipart Upload
- マルチパートアップロードの中止
- パートのリスト
- マルチパートアップロードのリスト

AWS CLI の使用

マルチパートアップロードのオペレーションについては、AWS Command Line Interface (AWS CLI) の以下のセクションを参照してください。

- Initiate Multipart Upload
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- マルチパートアップロードの中止
- パートのリスト
- マルチパートアップロードのリスト

REST API を使用して独自の REST リクエストを作成するか、AWS SDK のいずれかを使用することができます。REST API の詳細については、REST API の使用 を参照してください。SDK の詳細については、マルチパートアップロードを使用したオブジェクトのアップロード を参照してください。

高レベルの .NET TransferUtility クラスを使用してディレクトリをアップロードする

ディレクトリ全体をアップロードするには、`TransferUtility` クラスを使用できます。デフォルトでは、API でアップロードされるのは、指定したディレクトリのルートにあるファイルのみです。ただし、すべてのサブディレクトリでファイルを再帰的にアップロードするように指定できます。

フィルタ基準に基づいて指定したディレクトリのファイルを選択するには、フィルタ式を指定します。例えば、ディレクトリから .pdf ファイルのみをアップロードするには、`"*.pdf"` フィルタ式を指定します。

ディレクトリからファイルをアップロードする場合は、結果のオブジェクトのキー名を指定しません。キー名は、元のファイルパスを使用して Amazon S3 で作成されます。例えば、以下の構造の `c:\myfolder` ディレクトリがあるとします。

Example

```
C:\myfolder
      \a.txt
      \b.pdf
      \media\
             An.mp3
```

このディレクトリをアップロードすると、Amazon S3 では次のキー名が使用されます。

Example

```
a.txt
b.pdf
media/An.mp3
```

Example

以下の C# コード例では、ディレクトリを Amazon S3 バケットにアップロードしています。さまざまな `TransferUtility.UploadDirectory` オーバーロードを使用してディレクトリをアップロードする方法を示します。後続のアップロード呼び出しが行われるたびに、前のアップロードが置き換えられます。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadDirMPUHighLevelAPITest
    {
        private const string existingBucketName = "*** bucket name ***";
        private const string directoryPath = @"*** directory path ***";
        // The example uploads only .txt files.
```

```
        private const string wildCard = "*.txt";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            UploadDirAsync().Wait();
        }

        private static async Task UploadDirAsync()
        {
            try
            {
                var directoryTransferUtility =
                    new TransferUtility(s3Client);

                // 1. Upload a directory.
                await directoryTransferUtility.UploadDirectoryAsync(directoryPath,
                    existingBucketName);
                Console.WriteLine("Upload statement 1 completed");

                // 2. Upload only the .txt files from a directory
                //    and search recursively.
                await directoryTransferUtility.UploadDirectoryAsync(
                                              directoryPath,
                                              existingBucketName,
                                              wildCard,
                                              SearchOption.AllDirectories);
                Console.WriteLine("Upload statement 2 completed");

                // 3. The same as Step 2 and some optional configuration.
                //    Search recursively for .txt files to upload.
                var request = new TransferUtilityUploadDirectoryRequest
                {
                    BucketName = existingBucketName,
                    Directory = directoryPath,
                    SearchOption = SearchOption.AllDirectories,
                    SearchPattern = wildCard
                };

                await directoryTransferUtility.UploadDirectoryAsync(request);
                Console.WriteLine("Upload statement 3 completed");
            }
```

```
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an object",
 e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Unknown encountered on server. Message:'{0}' when writing an
 object", e.Message);
            }
        }
    }
}
```

マルチパートアップロードのリスト化

AWS SDK (低レベル API) を使用して、Amazon S3 で進行中のマルチパートアップロードのリストを取得できます。

AWS SDK (低レベル API) を使用したマルチパートアップロードのリスト化

Java

以下のタスクは、低レベル Java クラスを使用して、バケットで進行中のすべてのマルチパートアップロードをリストする手順を示しています。

低レベル API のマルチパートアップロードのリスト化プロセス

1	`ListMultipartUploadsRequest` クラスのインスタンスを作成し、バケット名を指定します。
2	`AmazonS3Client.listMultipartUploads` メソッドを実行します。このメソッドは、進行中のマルチパートアップロードに関する情報を提供する `MultipartUploadListing` クラスのインスタンスを返します。

以下の Java コード例は、前述のタスクの例です。

Example

```
ListMultipartUploadsRequest allMultpartUploadsRequest =
```

```
    new ListMultipartUploadsRequest(existingBucketName);
MultipartUploadListing multipartUploadListing =
    s3Client.listMultipartUploads(allMultpartUploadsRequest);
```

.NET

特定のバケットですべての進行中のマルチパートアップロードを一覧表示するには、AWS SDK for .NET の低レベルのマルチパートアップロード API の `ListMultipartUploadsRequest` クラスを使用します。`AmazonS3Client.ListMultipartUploads` メソッドは、`ListMultipartUploadsResponse` クラスのインスタンスを返し、進行中のマルチパートアップロードに関する情報を提供します。

進行中のマルチパートアップロードとは、マルチパートアップロードの開始リクエストによって開始されてから、まだ完了または中止されていないマルチパートアップロードです。Amazon S3 マルチパートアップロードの詳細については、マルチパートアップロードを使用したオブジェクトのアップロードとコピー を参照してください。

次の C# の例では、AWS SDK for .NET を使用してバケットで進行中のマルチパートアップロードを一覧表示する方法を示します。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
ListMultipartUploadsRequest request = new ListMultipartUploadsRequest
{
  BucketName = bucketName // Bucket receiving the uploads.
};

ListMultipartUploadsResponse response = await
 AmazonS3Client.ListMultipartUploadsAsync(request);
```

PHP

このトピックでは、バージョン 3 の AWS SDK for PHP の低レベル API クラスを使用して、バケットで進行中のすべてのマルチパートアップロードを一覧表示する方法を示します。AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

次の PHP の例では、バケットで進行中のすべてのマルチパートアップロードを一覧表示します。

```
require 'vendor/autoload.php';
```

```
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Retrieve a list of the current multipart uploads.
$result = $s3->listMultipartUploads([
    'Bucket' => $bucket
]);

// Write the list of uploads to the page.
print_r($result->toArray());
```

REST API を使用したマルチパートアップロードのリスト化

Amazon Simple Storage Service API リファレンスの以下のセクションでは、マルチパートアップロードをリスト化するための REST API について説明します。

- ListParts - 特定のマルチパートアップロードでアップロードされた部分を一覧表示します。
- ListMultipartUploads - 進行中のマルチパートアップロードを一覧表示します。

AWS CLI を使用したマルチパートアップロードのリスト化

マルチパートアップロードのオペレーションについては、AWS Command Line Interface の以下のセクションを参照してください。

- list-parts - 特定のマルチパートアップロードでアップロードされた部分を一覧表示します。
- list-multipart-uploads - 進行中のマルチパートアップロードを一覧表示します。

マルチパートアップロードの追跡

高レベルのマルチパートアップロード API は、Amazon S3 へのオブジェクトのアップロード時に進行状況を追跡するリッスンインターフェイス `ProgressListener` を備えています。進行状況に関するイベントが定期的に発生し、バイトが転送されたことをリスナーに通知します。

Java

Example

```
TransferManager tm = new TransferManager(new ProfileCredentialsProvider());

PutObjectRequest request = new PutObjectRequest(
    existingBucketName, keyName, new File(filePath));

// Subscribe to the event and provide event handler.
request.setProgressListener(new ProgressListener() {
   public void progressChanged(ProgressEvent event) {
    System.out.println("Transferred bytes: " +
      event.getBytesTransfered());
            }
});
```

Example

以下の Java コードは、ファイルをアップロードし、ProgressListener を使用してアップロードの進行状況を追跡します。作業サンプルの作成方法およびテスト方法については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import java.io.File;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.Upload;

public class TrackMPUProgressUsingHighLevelAPI {

    public static void main(String[] args) throws Exception {
        String existingBucketName = "*** Provide bucket name ***";
        String keyName            = "*** Provide object key ***";
        String filePath           = "*** file to upload ***";

        TransferManager tm = new TransferManager(new ProfileCredentialsProvider());
```

```
        // For more advanced uploads, you can create a request object
        // and supply additional request parameters (ex: progress listeners,
        // canned ACLs, etc.)
        PutObjectRequest request = new PutObjectRequest(
          existingBucketName, keyName, new File(filePath));

        // You can ask the upload for its progress, or you can
        // add a ProgressListener to your request to receive notifications
        // when bytes are transferred.
        request.setGeneralProgressListener(new ProgressListener() {
   @Override
   public void progressChanged(ProgressEvent progressEvent) {
    System.out.println("Transferred bytes: " +
      progressEvent.getBytesTransferred());
   }
  });

        // TransferManager processes all transfers asynchronously,
        // so this call will return immediately.
        Upload upload = tm.upload(request);

        try {
         // You can block and wait for the upload to finish
         upload.waitForCompletion();
        } catch (AmazonClientException amazonClientException) {
         System.out.println("Unable to upload file, upload aborted.");
         amazonClientException.printStackTrace();
        }
    }
}
```

.NET

次の C# の例では、TransferUtility クラスを使用して S3 バケットにファイルをアップロードし、アップロードの進行状況を追跡します。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.Threading.Tasks;
```

```
namespace Amazon.DocSamples.S3
{
    class TrackMPUUsingHighLevelAPITest
    {
        private const string bucketName = "*** provide the bucket name ***";
        private const string keyName = "*** provide the name for the uploaded object
 ***";
        private const string filePath = " *** provide the full path name of the file
 to upload **";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;


        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            TrackMPUAsync().Wait();
        }

        private static async Task TrackMPUAsync()
        {
            try
            {
                var fileTransferUtility = new TransferUtility(s3Client);

                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName
                    };

                uploadRequest.UploadProgressEvent +=
                    new EventHandler<UploadProgressArgs>
                        (uploadRequest_UploadPartProgressEvent);

                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
```

```
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }

        static void uploadRequest_UploadPartProgressEvent(object sender,
 UploadProgressArgs e)
        {
            // Process event.
            Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
        }
    }
}
```

マルチパートアップロードの中止

マルチパートアップロードを開始したら、パートをアップロードし始めます。Amazon S3 にはこれらのパートが保存されますが、それらのパートをすべてアップロードし、successful リクエストを送信してマルチパートアップロードを完了した場合のみオブジェクトが作成されます (マルチパートアップロードの完了リクエストが成功したことを確認してください)。マルチパートアップロードの完了リクエストを受け取ると、Amazon S3 はパートを組み立ててオブジェクトを作成します。マルチパートアップロードの完了リクエストが正常に送信されなかった場合、Amazon S3 はパートを組み立てず、オブジェクトも作成しません。

アップロードされたパートに関連のあるすべてのストレージに対して料金が請求されます。詳細については、「マルチパートアップロードと料金」を参照してください。そのため、マルチパートアップロードを完了してオブジェクトを作成するか、マルチパートアップロードを停止してアップロードされたパートを削除することが重要です。

AWS Command Line Interface (AWS CLI)、REST API、または AWS SDK を使用して、Amazon S3 で進行中のマルチパートアップロードを中止できます。バケットライフサイクル設定を使用して、完了しないマルチパートアップロードを中止することもできます。

AWS SDK (高レベル API) の使用

Java

TransferManager クラスは、進行中のマルチパートアップロードを中止する abortMultipartUploads メソッドを提供します。アップロードを開始すると、そのアップロードを完了または中止するまで進行中とみなされます。Date 値を指定すると、この API が、そのバケットで指定した Date の前に開始された進行中のすべてのマルチパートアップロードを中止します。

以下のタスクでは、高レベル Java クラスを使用してマルチパートアップロードを中止する手順を説明します。

高レベル API のマルチパートアップロード中止プロセス

1	TransferManager クラスのインスタンスを作成します。
2	バケット名および TransferManager.abortMultipartUploads 値を渡して Date メソッドを実行します。

次の Java コードでは、1 週間以上前に特定のバケットで開始された進行中のマルチパートアップロードをすべて中止します。作業サンプルの作成方法およびテスト方法については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import java.util.Date;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.transfer.TransferManager;

public class AbortMPUUsingHighLevelAPI {

    public static void main(String[] args) throws Exception {
        String existingBucketName = "*** Provide existing bucket name ***";

        TransferManager tm = new TransferManager(new ProfileCredentialsProvider());


        int sevenDays = 1000 * 60 * 60 * 24 * 7;
  Date oneWeekAgo = new Date(System.currentTimeMillis() - sevenDays);
```

```
        try {
         tm.abortMultipartUploads(existingBucketName, oneWeekAgo);
        } catch (AmazonClientException amazonClientException) {
         System.out.println("Unable to upload file, upload was aborted.");
         amazonClientException.printStackTrace();
        }
    }
}
```

> **Note**
>
> 特定のマルチパートアップロードを中止することもできます。詳細については、「AWS SDK の使用 (低レベル API)」を参照してください。

.NET

次の C# の例では、1 週間以上前に特定のバケットで開始された進行中のマルチパートアップロードをすべて中止します。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class AbortMPUUsingHighLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            AbortMPUAsync().Wait();
```

```
        }

        private static async Task AbortMPUAsync()
        {
            try
            {
                var transferUtility = new TransferUtility(s3Client);

                // Abort all in-progress uploads initiated before the specified
 date.
                await transferUtility.AbortMultipartUploadsAsync(
                    bucketName, DateTime.Now.AddDays(-7));
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

> **Note**
>
> 特定のマルチパートアップロードを中止することもできます。詳細については、「AWS SDK の使用 (低レベル API)」を参照してください。

AWS SDK の使用 (低レベル API)

`AmazonS3.abortMultipartUpload` メソッドを呼び出すと、進行中のマルチパートアップロードを中止できます。このメソッドは、Amazon S3 にアップロードされたすべてのパートを削除し、リソースを解放します。アップロード ID、バケット名、およびキー名を指定する必要があります。以下の Java コードの例では、進行中のマルチパートアップロードを中止する方法を示します。

マルチパートアップロードを中止するには、アップロード ID とアップロードで使用しているバケット名とキー名を指定します。マルチパートアップロードを中止した後は、同じアップロード ID を使

用して追加のパートをアップロードすることはできません。Amazon S3 マルチパートアップロードの詳細については、マルチパートアップロードを使用したオブジェクトのアップロードとコピー を参照してください。

Java

次の Java コード例では、進行中のマルチパートアップロードを中止します。

Example

```
InitiateMultipartUploadRequest initRequest =
    new InitiateMultipartUploadRequest(existingBucketName, keyName);
InitiateMultipartUploadResult initResponse =
                s3Client.initiateMultipartUpload(initRequest);

AmazonS3 s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
s3Client.abortMultipartUpload(new AbortMultipartUploadRequest(
            existingBucketName, keyName, initResponse.getUploadId()));
```

> **Note**
>
> 特定のマルチパートアップロードではなく、特定の時刻より前に開始された進行中のマルチパートアップロードをすべて中止することができます。このクリーンアップオペレーションは、開始したが完了または中止していない古いマルチパートアップロードを中止する場合に役に立ちます。詳細については、「AWS SDK (高レベル API) の使用」を参照してください。

.NET

次の C# の例は、マルチパートアップロードを中止する方法を示しています。次のコードを含む完全な C# のサンプルについては、AWS SDK の使用 (低レベル API) を参照してください。

```
AbortMultipartUploadRequest abortMPURequest = new AbortMultipartUploadRequest
{
    BucketName = existingBucketName,
    Key = keyName,
    UploadId = initResponse.UploadId
};
await AmazonS3Client.AbortMultipartUploadAsync(abortMPURequest);
```

特定の時刻までに開始されたすべての進行中のマルチパートアップロードを中止することもできます。このクリーンアップオペレーションは、完了または中止していないマルチパートアップロードを中止する場合に役立ちます。詳細については、「AWS SDK (高レベル API) の使用」を参照してください。

PHP

この例では、バージョン 3 の AWS SDK for PHP のクラスを使用して、進行中のマルチパートアップロードを中止する方法を示します。AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。`abortMultipartUpload()` メソッドの例です。

AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$uploadId = '*** Upload ID of upload to Abort ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Abort the multipart upload.
$s3->abortMultipartUpload([
    'Bucket'   => $bucket,
    'Key'      => $keyname,
    'UploadId' => $uploadId,
]);
```

REST API の使用

REST API を使用してマルチパートアップロードを中止する方法の詳細については、Amazon Simple Storage Service API リファレンスの AbortMultiPartUpload を参照してください。

AWS CLI の使用

AWS CLI を使用してマルチパートアップロードを中止する方法の詳細については、AWS CLI コマンドリファレンスの abort-multipart-upload を参照してください。

マルチパートアップロードを使用したオブジェクトのコピー

このセクションの例は、Multipart Upload API を使用して 5 GB よりも大きいオブジェクトをコピーする方法を示しています。5 GB よりも小さいオブジェクトは 1 回のオペレーションでコピーできます。詳細については、「オブジェクトのコピー、移動、名前の変更」を参照してください。

AWS SDK の使用

低レベル API を使用してオブジェクトをコピーするには、次の手順を実行します。

- `AmazonS3Client.initiateMultipartUpload()` メソッドを呼び出して、マルチパートアップロードを開始します。
- `AmazonS3Client.initiateMultipartUpload()` メソッドから返されたレスポンスオブジェクトのアップロード ID を保存します。このアップロード ID は、パートのアップロードオペレーションごとに指定します。
- すべてのパートをコピーします。コピーする必要があるパートごとに、`CopyPartRequest` クラスの新しいインスタンスを作成します。パート情報として、送信元と送信先のバケット名、送信元と送信先のオブジェクトキー、アップロード ID、パートの最初と最後のバイトの場所、パート番号などを指定します。
- `AmazonS3Client.copyPart()` メソッド呼び出しのレスポンスを保存します。各レスポンスには、アップロードしたパートの `ETag` 値とパート番号が含まれています。この情報は、マルチパートアップロードを完了するために必要です。
- `AmazonS3Client.completeMultipartUpload()` メソッドを呼び出してコピーオペレーションを完了します。

Java

Example

次の例では、Amazon S3 の下位 Java API を使用してマルチパートコピーを実行する方法を示します。作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class LowLevelMultipartCopy {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String sourceBucketName = "*** Source bucket name ***";
        String sourceObjectKey = "*** Source object key ***";
        String destBucketName = "*** Target bucket name ***";
        String destObjectKey = "*** Target object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(destBucketName,
                    destObjectKey);
            InitiateMultipartUploadResult initResult =
 s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest(sourceBucketName, sourceObjectKey);
            ObjectMetadata metadataResult =
 s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();

            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
```

```
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make
sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize -
1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(sourceBucketName)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(destBucketName)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and
make the
            // copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
                    destBucketName,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
```

```
    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
    }
}
```

.NET

次の C# の例では、AWS SDK for .NET を使用して 5 GB を超える Amazon S3 オブジェクトをコピー元からコピー先 (あるバケットから別のバケットなど) にコピーする方法を示します。5 GB 未満のオブジェクトをコピーするには、1 回のオペレーションでコピーする手順を使用します (AWS SDK の使用 を参照)。Amazon S3 マルチパートアップロードの詳細については、マルチパートアップロードを使用したオブジェクトのアップロードとコピー を参照してください。

この例では、AWS SDK for .NET マルチパートアップロード API を使用して、5 GB を超える Amazon S3 オブジェクトを S3 バケットから別のバケットにコピーする方法を示します。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CopyObjectUsingMPUapiTest
    {
        private const string sourceBucket = "*** provide the name of the bucket with
 source object ***";
        private const string targetBucket = "*** provide the name of the bucket to
 copy the object to ***";
        private const string sourceObjectKey = "*** provide the name of object to
 copy ***";
        private const string targetObjectKey = "*** provide the name of the object
 copy ***";
        // Specify your bucket region (an example region is shown).
```

```
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
            MPUCopyObjectAsync().Wait();
        }
        private static async Task MPUCopyObjectAsync()
        {
            // Create a list to store the upload part responses.
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();
            List<CopyPartResponse> copyResponses = new List<CopyPartResponse>();

            // Setup information required to initiate the multipart upload.
            InitiateMultipartUploadRequest initiateRequest =
                new InitiateMultipartUploadRequest
                {
                    BucketName = targetBucket,
                    Key = targetObjectKey
                };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            String uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = sourceBucket,
                    Key = sourceObjectKey
                };

                GetObjectMetadataResponse metadataResponse =
                    await s3Client.GetObjectMetadataAsync(metadataRequest);
```

```
                long objectSize = metadataResponse.ContentLength; // Length in
bytes.

                // Copy the parts.
                long partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    CopyPartRequest copyRequest = new CopyPartRequest
                    {
                        DestinationBucket = targetBucket,
                        DestinationKey = targetObjectKey,
                        SourceBucket = sourceBucket,
                        SourceKey = sourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i
                    };

                    copyResponses.Add(await s3Client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
                }

                // Set up to complete the copy.
                CompleteMultipartUploadRequest completeRequest =
                new CompleteMultipartUploadRequest
                {
                    BucketName = targetBucket,
                    Key = targetObjectKey,
                    UploadId = initResponse.UploadId
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (AmazonS3Exception e)
            {
```

```
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

REST API の使用

Amazon Simple Storage Service API リファレンスの以下のセクションでは、マルチパートアップロードの REST API について説明しています。既存のオブジェクトをコピーするには、UploadPart (Copy) API を使用し、リクエストに `x-amz-copy-source` リクエストヘッダーを追加してコピー元オブジェクトを指定します。

- Initiate Multipart Upload
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- マルチパートアップロードの中止
- パートのリスト
- マルチパートアップロードのリスト

これらの API を使用して独自の REST リクエストを作成するか、提供されている SDK のいずれかを使用できます。AWS CLI でマルチパートアップロードを使用する方法の詳細については、AWS CLI の使用 を参照してください。SDK の詳細については、マルチパートアップロードの AWS SDK サポート を参照してください。

Amazon S3 マルチパートアップロードの制限

次の表は、マルチパートアップロードの主な仕様をまとめたものです。詳細については、「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。

項目	仕様
最大オブジェクトサイズ	5 TiB
アップロードあたりの最大パート数	10,000
パート番号	1 ~ 10,000
パートサイズ	5 MiB から 5 GiB。マルチパートアップロードの最後のパートには、最小サイズの制限はありません。
パートのリストリクエストで返されるパートの最大数	1000
マルチパートアップロードのリストリクエストで返されるマルチパートアップロードの最大数	1000

オブジェクトのコピー、移動、名前の変更

CopyObject オペレーションを使用すると、Amazon S3 内に既に保存されているオブジェクトのコピーを作成できます。

1 回のアトミックオペレーションでコピーできるオブジェクトのサイズは最大 5 GB です。ただし、5 GB を超えるオブジェクトをコピーする場合は、マルチパートアップロード API を使用する必要があります。詳細については、「the section called “オブジェクトのコピー”」を参照してください。

CopyObject オペレーションを使用すると、以下のことができます。

- オブジェクトの追加のコピーを作成します。
- オブジェクトをコピーし、元のオブジェクトを削除して、オブジェクトの名前を変更します。
- 複数の AWS リージョン 間 (us-west-1 から eu-west-2 へなど) を含め、オブジェクトをあるバケットから別のバケットにコピーしたり、移動したりします。オブジェクトを移動する場合、Amazon S3 はオブジェクトを指定されたコピー先にコピーしてから、コピー元のオブジェクトを削除します。

> **Note**
>
> AWS リージョン 間でオブジェクトをコピーすると、帯域コストが発生します。詳細については、Amazon S3 の料金 を参照してください。

- オブジェクトのメタデータを変更する 各 Amazon S3 オブジェクトにはメタデータがあります。このメタデータは、名前と値のペアのセットです。オブジェクトメタデータは、オブジェクトをアップロードする際に設定できます。オブジェクトのアップロード後にオブジェクトメタデータを変更することはできません。オブジェクトメタデータを変更する唯一の方法は、オブジェクトのコピーを作成し、メタデータを設定することです。これを行うには、コピーオペレーションで、コピー元とコピー先として同じオブジェクトを指定します。

  オブジェクトのメタデータによっては、システムメタデータであったり、ユーザー定義メタデータである場合もあります。システムメタデータによっては、制御できます。例えば、オブジェクトに使用するストレージクラスやサーバー側の暗号化の種類を制御できます。オブジェクトをコピーするときは、ユーザーが制御するシステムメタデータとユーザー定義メタデータもコピーされます。システムが制御するメタデータは Amazon S3 によってリセットされます。たとえば、オブジェクトをコピーすると、コピーしたオブジェクトの作成日が Amazon S3 によってリセットされます。コピーリクエストでは、このようなシステム制御のメタデータ値を設定する必要はありません。

  オブジェクトをコピーするときは、一部のメタデータ値を更新することができます。例えば、コピー元のオブジェクトが S3 標準ストレージを使用するように設定されている場合、オブジェクトコピーに S3 Intelligent-Tiering を使用することができます。また、コピー元のオブジェクトに存在するユーザー定義メタデータの一部を変更することもできます。オブジェクトのユーザー設定可能なメタデータ (システムまたはユーザー定義) をコピー時に更新する場合、メタデータ値の 1 つだけを変更する場合でも、コピー元オブジェクトに存在するすべてのユーザー設定可能メタデータをリクエスト内で明示的に指定する必要があります。

  オブジェクトメタデータの詳細については、オブジェクトメタデータの使用 を参照してください。

アーカイブされたオブジェクトや復元されたオブジェクトのコピー

ソースオブジェクトが S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive にアーカイブされている場合は、オブジェクトを他のバケットにコピーする前に、一時コピーを復元する必要があります。オブジェクトのアーカイブについては、S3 Glacier Flexible Retrieval と S3 Glacier Deep Archive ストレージクラスへの移行 (オブジェクトのアーカイブ) を参照してください。

Amazon S3 コンソールのコピーオペレーションは、S3 Glacier Flexible Retrieval や S3 Glacier Deep Archive ストレージクラスの復元されたオブジェクトではサポートされていません。復元されたこのようなオブジェクトをコピーするには、AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 REST API を使用します。

暗号化されたオブジェクトのコピー

Amazon S3 は、S3 バケットにコピーされたすべての新しいオブジェクトを自動的に暗号化します。コピーリクエストで暗号化情報を指定しない場合、ターゲットオブジェクトの暗号化設定は送信先バケットのデフォルトの暗号化設定に設定されます。デフォルトでは、すべてのバケットには、Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) を使用します。送信先バケットに AWS Key Management Service (AWS KMS) キー (SSE-KMS) によるサーバー側の暗号化またはお客様が指定した暗号化キー (SSE-C) を使用するデフォルトの暗号化設定がある場合、Amazon S3 は対応する KMS キーまたはお客さんが指定したキーを使用してターゲットオブジェクトのコピーを暗号化します。

オブジェクトをコピーする際、ターゲットオブジェクトに別のタイプの暗号化設定を使用する場合は、Amazon S3 が KMS キー、Amazon S3 マネージドキー、またはお客様が提供するキーを使用してターゲットオブジェクトを暗号化するようにリクエストできます。リクエストの暗号化設定が送信先バケットのデフォルトの暗号化設定と異なる場合は、リクエストの暗号化設定が優先されます。コピー元のオブジェクトが SSE-C を使用して暗号化されている場合、Amazon S3 がコピーするオブジェクトを復号化できるように、リクエストで必要な暗号化情報を指定する必要があります。詳細については、「暗号化によるデータの保護」を参照してください。

オブジェクトコピー時のチェックサムの使用

オブジェクトをコピーするとき、オブジェクトに対して別のチェックサムアルゴリズムを使用するように選択できます。同じアルゴリズムを使用するか、新しいアルゴリズムを使用するかにかかわらず、Amazon S3 はオブジェクトのコピー後に新しいチェックサム値を計算します。Amazon S3 は、チェックサムの値を直接コピーしません。マルチパートアップロードを使用してロードされたオブジェクトのチェックサム値は変更される場合があります。チェックサムの計算方法の詳細については、「マルチパートアップロードにパートレベルのチェックサムを使用する」を参照してください。

単一のリクエストでの複数オブジェクトのコピー

単一のリクエストで複数の Amazon S3 オブジェクトをコピーするには、Amazon S3 バッチオペレーションを使用することもできます。S3 バッチオペレーションには、オペレーションターゲットのオブジェクトのリストを指定します。S3 バッチオペレーションは、各 API オペレーションを呼び出して、指定されたオペレーションを実行します。1 つのバッチオペレーションジョブで、エクサバイトのデータを含む数十億ものオブジェクトに対して、指定されたオペレーションを実行できます。

S3 バッチオペレーション機能は、進捗状況の追跡、通知の送信、すべてのアクションの詳細な完了レポートの保存を行い、フルマネージドの監査可能なサーバーレスエクスペリエンスを提供します。S3 バッチオペレーションは、Amazon S3 コンソール、AWS CLI、AWS SDK、または REST API を通じて使用できます。詳細については、「the section called “バッチ操作の基本”」を参照してください。

オブジェクトのディレクトリバケットへのコピー

オブジェクトをディレクトリバケットにコピーする方法の詳細については、「オブジェクトのディレクトリバケットへのコピー」を参照してください。Amazon S3 Express One Zone ストレージ クラスをディレクトリバケットで使用する方法については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

オブジェクトをコピーするには

オブジェクトをコピーするには、以下のメソッドを使用します。

S3 コンソールの使用

> **Note**
>
> - Amazon S3 コンソールを使用してオブジェクトをコピーする場合は、アクセス許可 `s3:ListAllMyBuckets` が付与されている必要があります。コンソールには、コピーオペレーションを検証するうえで、このアクセス許可が必要です。このアクセス許可を付与するポリシーの例については、「the section called “アイデンティティベースポリシーの例”」を参照してください。
>
>   ユーザー定義のタグを持つオブジェクトをコピーする場合は、`s3:GetObjectTagging` アクセス許可も必要です。ユーザー定義タグがなく、サイズが 16 MB を超えるオブジェクトをコピーする場合は、s3:GetObjectTagging アクセス許可も必要です。
>
>   ターゲットバケットポリシーによって `s3:GetObjectTagging` アクションが拒否されると、ユーザー定義タグなしでオブジェクトがコピーされ、エラーが発生します。
> - お客様が提供する暗号化キー (SSE-C) で暗号化されたオブジェクトは、S3 コンソールを使用してコピーすることはできません。SSE-C で暗号化されたオブジェクトをコピーするには、AWS CLI、AWS SDK、または Amazon S3 REST API を使用します。
> - SSE-KMS で暗号化されたオブジェクトのクロスリージョンコピーは、Amazon S3 コンソールではサポートされていません。SSE-KMS で暗号化されたオブジェクトをリージョ

ン間でコピーするには、AWS CLI、AWS SDK、または Amazon S3 REST API を使用します。

オブジェクトをコピーするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで [バケット] を選択してから、[汎用バケット] タブをクリックします。コピーするオブジェクトを含む Amazon S3 バケットまたはフォルダに移動します。
3. コピーするオブジェクトの名前の左にあるチェックボックスをオンにします。
4. [アクション] メニューに表示されるオプションのリストから [コピー] を選択します。
5. 送信先タイプと送信先アカウントを選択します。送信先パスを指定するには、[S3 の参照] を選択し、送信先に移動して、送信先の左側にあるチェックボックスをオンにします。右下の [Choose destination] (送信先を選択する) を選択します。

   または、送信先パスを入力します。
6. バケットのバージョニングが有効になっていない場合は、同じ名前の既存のオブジェクトが上書きされることを確認するように求められる場合があります。上書きされてもよい場合は、チェックボックスを選択して続行します。このバケットのオブジェクトの全バージョンを保持する場合は、[バケットのバージョニングを有効にする] をクリックします。デフォルトの暗号化プロパティと S3 オブジェクトロックプロパティを更新することもできます。
7. [Additional checksums] (追加のチェックサム) で、既存のチェックサム関数を使用してオブジェクトをコピーするか、既存のチェックサム関数を新しいチェックサム関数に置き換えるかを選択します。オブジェクトをアップロードしたとき、データの整合性を検証するために使用されたチェックサムアルゴリズムを指定するオプションがありました。オブジェクトをコピーするとき、新しい関数を選択するオプションがあります。最初に追加のチェックサムを指定しなかった場合は、コピーオプションのこのセクションを使用して追加できます。

   > **Note**
   >
   > 同じチェックサム関数を使用することを選択した場合でも、オブジェクトをコピーして、サイズが 16 MB を超えると、チェックサム値が変更されることがあります。チェックサム値は、マルチパートアップロードのチェックサムの計算方法によって変わる可能性があります。オブジェクトをコピーするときのチェックサムの変更については、「マ

ルチパートアップロードにパートレベルのチェックサムを使用する」を参照してください。

チェックサム関数を変更するには、[Replace with a new checksum function] (新しいチェックサム関数で置き換える) を選択します。ボックスから新しいチェックサム関数を選択します。オブジェクトがコピーされると、指定されたアルゴリズムを使用して新しいチェックサムが計算され、保存されます。

8. 右下の [Copy] (コピー) を選択します。Amazon S3 によってオブジェクトがコピー先にコピーされます。

AWS SDK の使用

このセクションの例は、1 回のオペレーションで最大 5 GB のオブジェクトをコピーする方法を示しています。5 GB を超えるオブジェクトをコピーする場合は、マルチパートアップロード API を使用する必要があります。詳細については、「マルチパートアップロードを使用したオブジェクトのコピー」を参照してください。

Java

Example

次の例では、AWS SDK for Java を使用して Amazon S3 にオブジェクトをコピーします。作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;

import java.io.IOException;

public class CopyObjectSingleOperation {

    public static void main(String[] args) throws IOException {
```

```
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String sourceKey = "*** Source object key *** ";
        String destinationKey = "*** Destination object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Copy the object into a new object in the same bucket.
            CopyObjectRequest copyObjRequest = new CopyObjectRequest(bucketName,
 sourceKey, bucketName, destinationKey);
            s3Client.copyObject(copyObjRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

次の C# の例では、高レベルの AWS SDK for .NET を使用して、1 回のオペレーションで 5 GB の大きさのオブジェクトをコピーします。5 GB を超えるオブジェクトには、マルチパートアップロードを使用したオブジェクトのコピー で説明しているマルチパートアップロードコピーの例を使用します。

この例では、最大 5 GB のオブジェクトのコピーを作成します。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
```

```
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CopyObjectTest
    {
        private const string sourceBucket = "*** provide the name of the bucket with
 source object ***";
        private const string destinationBucket = "*** provide the name of the bucket
 to copy the object to ***";
        private const string objectKey = "*** provide the name of object to copy
 ***";
        private const string destObjectKey = "*** provide the destination object key
 name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
            CopyingObjectAsync().Wait();
        }

        private static async Task CopyingObjectAsync()
        {
            try
            {
                CopyObjectRequest request = new CopyObjectRequest
                {
                    SourceBucket = sourceBucket,
                    SourceKey = objectKey,
                    DestinationBucket = destinationBucket,
                    DestinationKey = destObjectKey
                };
                CopyObjectResponse response = await
 s3Client.CopyObjectAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
```

```
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

このトピックでは、バージョン 3 の AWS SDK for PHP のクラスを使用して、Amazon S3 内の 1 つまたは複数のオブジェクトを、特定のバケットから別のバケットへ、または同じバケット内にコピーする手順を示します。

AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

以下の PHP の例は、copyObject() メソッドを使用して Amazon S3 内の単一つのオブジェクトをコピーする方法を説明しています。getcommand() メソッドを使用して、CopyObject 呼び出しのバッチを使用して、オブジェクトの複数コピーを作成する例も示します。

オブジェクトのコピー

1	Aws\S3\S3Client クラスのコンストラクタを使用して、Amazon S3 クライアントのインスタンスを作成します。
2	オブジェクトの複数のコピーを作成するには、Amazon S3 クライアントの getCommand() メソッドへの呼び出しのバッチを実行します。このメソッドは、Aws\CommandInterface クラスから継承されています。最初の引数として CopyObject コマンドを指定し、2 番目の引数としてコピー元のバケット、コピー元のキー名、コピー先のバケット、コピー先のキー名を含む array を指定します。

```
 require 'vendor/autoload.php';

use Aws\CommandPool;
use Aws\Exception\AwsException;
use Aws\ResultInterface;
```

```
use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
$sourceKeyname = '*** Your Source Object Key ***';
$targetBucket = '*** Your Target Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

// Copy an object.
$s3->copyObject([
    'Bucket' => $targetBucket,
    'Key' => "$sourceKeyname-copy",
    'CopySource' => "$sourceBucket/$sourceKeyname",
]);

// Perform a batch of CopyObject operations.
$batch = array();
for ($i = 1; $i <= 3; $i++) {
    $batch[] = $s3->getCommand('CopyObject', [
        'Bucket' => $targetBucket,
        'Key' => "{targetKeyname}-$i",
        'CopySource' => "$sourceBucket/$sourceKeyname",
    ]);
}
try {
    $results = CommandPool::batch($s3, $batch);
    foreach ($results as $result) {
        if ($result instanceof ResultInterface) {
            // Result handling here
        }
        if ($result instanceof AwsException) {
            // AwsException handling here
        }
    }
} catch (Exception $e) {
    // General error handling here
}
```

Python

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key
```

```
    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
 key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
 self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
```

```
            )
            raise
```

Ruby

以下のタスクでは、Ruby クラスを使用して、Amazon S3 内のバケット間または同一バケット内でオブジェクトをコピーする方法を説明します。

オブジェクトのコピー

1	AWS SDK for Ruby のバージョン 3 には Amazon S3 のモジュール化された gem を使用し、aws-sdk-s3 を要求し、AWS 認証情報を指定します。認証情報の指定の詳細については、AWS アカウント または IAM ユーザーの認証情報を使用したリクエストの実行 を参照してください。
2	コピー元のバケット名、コピー元のキー名、コピー先のバケット名、コピー先のキー名などのリクエスト情報を指定します。

次の Ruby サンプルコードは、前のタスクについて、#copy_object メソッドを使用して、バケット間でオブジェクトをコピーする方法を説明しています。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
```

```
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

REST API の使用

この例では、Amazon S3 REST API を使用してオブジェクトをコピーする方法について説明します。REST API の詳細については、CopyObject を参照してください。

このサンプルは、flotsam オブジェクト (*example-s3-bucket1* バケット) を jetsam オブジェクト (*example-s3-bucket2* バケット) にコピーし、そのメタデータを維持します。

```
PUT /jetsam HTTP/1.1
Host: example-s3-bucket2.s3.amazonaws.com
x-amz-copy-source: /example-s3-bucket1/flotsam
Authorization: AWS AKIAIOSFODNN7EXAMPLE:ENoSbxYByFA0UGLZUqJN5EUnLDg=
```

```
Date: Wed, 20 Feb 2008 22:12:21 +0000
```

署名は次の情報から生成されています。

```
PUT\r\n
\r\n
\r\n
Wed, 20 Feb 2008 22:12:21 +0000\r\n

x-amz-copy-source:/example-s3-bucket1/flotsam\r\n
/example-s3-bucket2/jetsam
```

Amazon S3 から、オブジェクトの ETag と最終変更日を示す次のレスポンスが返されます。

```
HTTP/1.1 200 OK
x-amz-id-2: Vyaxt7qEbzv34BnSu5hctyyNSlHTYZFMWK4FtzO+iX8JQNyaLdTshL0KxatbaOZt
x-amz-request-id: 6B13C3C5B34AF333
Date: Wed, 20 Feb 2008 22:13:01 +0000

Content-Type: application/xml
Transfer-Encoding: chunked
Connection: close
Server: AmazonS3
<?xml version="1.0" encoding="UTF-8"?>

<CopyObjectResult>
   <LastModified>2008-02-20T22:13:01</LastModified>
   <ETag>"7e9c608af58950deeb370c98608ed097"</ETag>
</CopyObjectResult>
```

AWS CLI の使用

また、AWS Command Line Interface (AWS CLI) を使用して、S3 バケットをコピーすることもできます。詳細については、AWS CLI コマンドリファレンスの「copy-object」を参照してください。

AWS CLI の詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS Command Line Interface とは」を参照してください。

オブジェクトを移動するには

オブジェクトを移動するには、次の方法を使用します。

S3 コンソールの使用

> **Note**
>
> - ユーザー定義のタグを持つオブジェクトを移動する場合は、`s3:GetObjectTagging` アクセス許可が必要です。ユーザー定義タグがなく、サイズが 16 MB を超えるオブジェクトを移動する場合は、`s3:GetObjectTagging` アクセス許可も必要です。
>
> ターゲットバケットポリシーによって `s3:GetObjectTagging` アクションが拒否されると、ユーザー定義タグなしでオブジェクトが移動され、エラーが発生します。
> - お客様が提供する暗号化キー (SSE-C) で暗号化されたオブジェクトは、Amazon S3 コンソールを使用して移動することはできません。SSE-C で暗号化されたオブジェクトを移動するには、AWS CLI、AWS SDK、または Amazon S3 REST API を使用します。
> - フォルダを移動する場合、フォルダをさらに変更するには、移動オペレーションが完了するまで待ちます。
> - Amazon S3 コンソールの 移動オペレーションの移動元または移動先として S3 アクセスポイントのエイリアスを使用することはできません。

オブジェクトを移動するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで [バケット] を選択してから、[汎用バケット] タブをクリックします。移動するオブジェクトが含まれている Amazon S3 のバケットまたはフォルダに移動します。
3. 移動するオブジェクトの名前の左にあるチェックボックスをオンにします。
4. [アクション] メニューで、[移動] を選択します。
5. 送信先パスを指定するには、[S3 の参照] を選択し、送信先に移動して、送信先の左側にあるチェックボックスをオンにします。右下の [Choose destination] (送信先を選択する) を選択します。

   または、送信先パスを入力します。
6. バケットのバージョニングが有効になっていない場合は、同じ名前の既存のオブジェクトが上書きされることを確認するように求められる場合があります。上書きされてもよい場合は、チェックボックスを選択して続行します。このバケットのオブジェクトの全バージョンを保持する場

合は、[バケットのバージョニングを有効にする] をクリックします。デフォルトの暗号化プロパティとオブジェクトロックプロパティを更新することもできます。

7. 右下の [Move] (移動) を選択します。Amazon S3 は、オブジェクトを移動先に移動します。

> **Note**
>
> - このアクションでは、更新された設定で指定されたすべてのオブジェクトのコピーが作成されます。また、指定された場所の最終更新日が更新され、元のオブジェクトに削除マーカーが追加されます。
> - このアクションは、バケットのバージョニング、暗号化、オブジェクトロック機能、アーカイブされたオブジェクトのメタデータを更新します。

AWS CLI の使用

また、AWS Command Line Interface (AWS CLI) を使用して、S3 バケットを移動することもできます。詳細については、AWS CLI コマンドリファレンスの「mv」を参照してください。

AWS CLI の詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS Command Line Interface とは」を参照してください。

オブジェクトの名前を変更するには

オブジェクトの名前を変更するには、以下の手順を使用します。

> **Note**
>
> - オブジェクトの名前を変更すると、オブジェクトのコピーが新しい最終更新日で作成され、元のオブジェクトに削除マーカーが追加されます。
> - デフォルトの暗号化のバケット設定が、暗号化されていない指定されたオブジェクトに自動的に適用されます。
> - Amazon S3 コンソールを使用して、顧客が提供した暗号化キー (SSE-C) を使用してオブジェクトの名前を変更することはできません。SSE-C で暗号化されたオブジェクトの名前を変更するには、AWS CLI、AWS SDK、または Amazon S3 REST API を使用して、新しい名前でこれらのオブジェクトをコピーします。

- このバケットが S3 オブジェクトの所有権のバケット所有者の強制設定を使用している場合、オブジェクトアクセスコントロールリスト (ACL) はコピーされません。
- ユーザー定義のタグを持つオブジェクトの名前を変更する場合は、`s3:GetObjectTagging` アクセス許可が必要です。ユーザー定義タグがなく、サイズが 16 MB を超えるオブジェクトの名前を変更する場合は、`s3:GetObjectTagging` アクセス許可も必要です。

  ターゲットバケットポリシーによって `s3:GetObjectTagging` アクションが拒否されると、オブジェクトの名前は変更されますが、ユーザー定義タグはオブジェクトから削除され、エラーが発生します。

オブジェクトの名前を変更するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで [バケット] を選択してから、[汎用バケット] タブをクリックします。名前を変更するオブジェクトを含む Amazon S3 バケットまたはフォルダに移動します。
3. 名前を変更するオブジェクトの名前の左にあるチェックボックスをオンにします。
4. [アクション] メニューで [オブジェクトの名前変更] を選択します。
5. [新しいオブジェクト名] ボックスに、オブジェクトの新しい名前を入力します。
6. 右下の [変更の保存] を選択します。Amazon S3 によりオブジェクトの名前が変更されます。

オブジェクトのダウンロード

このセクションでは、Amazon S3 バケットからオブジェクトをダウンロードする方法について説明します。Amazon S3 では、オブジェクトを 1 つ以上のバケットに保存できます。各オブジェクトの最大サイズは 5 TB です。アーカイブされていない Amazon S3 オブジェクトには、リアルタイムでアクセスできます。ただし、オブジェクトがアーカイブされている場合は、ダウンロードする前に復元する必要があります。アーカイブされたオブジェクトのダウンロードの詳細については、「the section called “アーカイブされたオブジェクトのダウンロード”」を参照してください。

Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して 1 つのオブジェクトをダウンロードできます。コードを記述したり、コマンドを実行したりせずに、S3 からオブジェクトをダウンロードするには、S3 コンソールを使用します。詳細については、「the section called “オブジェクトのダウンロード”」を参照してください。

複数のオブジェクトをダウンロードするには、AWS CloudShell、AWS CLI、または AWS SDK を使用します。詳細については、「the section called “複数のオブジェクトのダウンロード”」を参照してください。

オブジェクトの一部をダウンロードする必要がある場合は、AWS CLI または REST API で追加のパラメータを使用して、ダウンロードするバイトのみを指定します。詳細については、「the section called “オブジェクトの一部のダウンロード”」を参照してください。

所有していないオブジェクトをダウンロードする必要がある場合は、オブジェクトの所有者に、そのオブジェクトのダウンロードを許可する署名付き URL を生成するよう依頼します。詳細については、「the section called “別の AWS アカウントに属するオブジェクトのダウンロード”」を参照してください。

AWS ネットワーク外のオブジェクトをダウンロードする場合は、データ転送料金が適用されます。AWS ネットワーク内でのデータ転送は、同じ AWS リージョン内では無料ですが、GET リクエストに対しては料金がかかります。データ転送コストとデータ取得料金の詳細については、「Amazon S3 の料金」を参照してください。

トピック

- オブジェクトのダウンロード
- 複数のオブジェクトのダウンロード
- オブジェクトの一部のダウンロード
- 別の AWS アカウントに属するオブジェクトのダウンロード
- アーカイブされたオブジェクトのダウンロード
- オブジェクトのダウンロードに関するトラブルシューティング

オブジェクトのダウンロード

Amazon S3 コンソール、AWS CLI、AWS SDK、または REST API を使用してオブジェクトをダウンロードできます。

S3 コンソールの使用

このセクションでは、Amazon S3 コンソールを使用して S3 バケットからオブジェクトをダウンロードする方法について説明します。

> **Note**
>
> - 一度にダウンロードできるオブジェクトは 1 つだけです。
> - Amazon S3 コンソールを使用して、キー名がピリオド (.) で終わるオブジェクトをダウンロードすると、ダウンロードしたオブジェクトのキー名からピリオドが削除されます。ダウンロードしたオブジェクトの名前の末尾のピリオドを保持するには、AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 REST API を使用する必要があります。

S3 バケットからオブジェクトをダウンロードするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets] (バケット) リストで、オブジェクトのダウンロード元になるバケット名を選択します。
3. 次のいずれかの方法で、S3 バケットからオブジェクトをダウンロードできます。
   - オブジェクトの横にあるチェックボックスを選択し、[ダウンロード] を選択します。オブジェクトを特定のフォルダにダウンロードする場合は、[アクション] メニューの [名前を付けてダウンロード] を選択します。
   - オブジェクトの特定のバージョンをダウンロードする場合は、[バージョンの表示] ボタンを選択します。目的のオブジェクトのバージョンの横にあるチェックボックスをオンにして、[ダウンロード] を選択します。オブジェクトを特定のフォルダにダウンロードする場合は、[アクション] メニューの [名前を付けてダウンロード] を選択します。

AWS CLI の使用

次の `get-object` コマンド例は、AWS CLI を使用して Amazon S3 からオブジェクトをダウンロードする方法を示しています。このコマンドは、バケット *example-s3-bucket1* からオブジェクト *folder/my_image* を取得します。オブジェクトは、*my_downloaded_image* という名前のファイルにダウンロードされます。

```
aws s3api get-object --bucket example-s3-bucket1 --key folder/
my_image my_downloaded_image
```

詳細と例については、「AWS CLI コマンドリファレンス」の「get-object」を参照してください。

AWS SDK の使用

AWS SDK を使用してオブジェクトをダウンロードする方法の例については、「AWS SDK または CLI で GetObject を使用する」を参照してください。

さまざまな AWS SDK の使用に関する一般的な情報については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。

REST API の使用

REST API を使用して Amazon S3 からオブジェクトを取得できます。詳細については、「Amazon Simple Storage Service API リファレンス」の「GetObject」を参照してください。

複数のオブジェクトのダウンロード

AWS CloudShell、AWS CLI、または AWS SDK を使用して複数のオブジェクトをダウンロードできます。

AWS Management Console で AWS CloudShell を使用する

AWS CloudShell はブラウザベースの事前に認証されたシェルで、AWS Management Console から直接起動できます。

AWS CloudShell の詳細については、「AWS CloudShell ユーザーガイド」の「CloudShell とは」を参照してください。

> **Important**
>
> AWS CloudShell の場合、ホームディレクトリのストレージは AWS リージョンごとに最大 1 GB です。そのため、合計がこの量を超えるオブジェクトとバケットを同期することはできません。その他の制限については、「AWS CloudShell ユーザーガイド」の「サービスクォータと制限」を参照してください。

AWS CloudShell を使用してオブジェクトをダウンロードするには

1. AWS Management Console にサインインして、CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 次のコマンドを実行して、バケット内のオブジェクトを CloudShell に同期します。次のコマンドでは、*example-s3-bucket1* という名前のバケットのオブジェクトを同期し、CloudShell

に *temp* という名前のフォルダを作成します。CloudShell はオブジェクトをこのフォルダに同期します。このコマンドを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3 sync s3://example-s3-bucket1 ./temp
```

> **Note**
>
> パターンマッチングを実行して特定のオブジェクトを除外するか含めるには、`--exclude "value"` パラメータや `--include "value"` パラメータを sync コマンドで使用できます。

3. 次のコマンドを実行して、*temp* という名前のフォルダのオブジェクトを、*temp.zip* という名前のファイルに圧縮します。

```
zip temp.zip -r temp/
```

4. [アクション] を選択し、[ファイルをダウンロード] を選択します。
5. ファイル名として「**temp.zip**」と入力し、[ダウンロード] を選択します。
6. (オプション) CloudShell で *temp.zip* ファイルと *temp* フォルダに同期されているオブジェクトを削除します。AWS CloudShell の場合、永続的ストレージは AWS リージョンごとに最大 1 GB です。

   次のコマンド例を使用して、.zip ファイルとフォルダを削除できます。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
rm temp.zip && rm -rf temp/
```

AWS CLI の使用

次の例では、AWS CLI を使用して、指定したディレクトリまたはプレフィックスの下にあるすべてのファイルまたはオブジェクトをダウンロードする方法を示しています。このコマンドは、バケット *example-s3-bucket1* 内のすべてのオブジェクトを現在のディレクトリにコピーします。このコマンド例を使用するには、*example-s3-bucket1* の代わりにバケット名を使用します。

```
aws s3 cp s3://example-s3-bucket1 . --recursive
```

次のコマンドは、バケット *example-s3-bucket1* 内のプレフィックス *logs* の下にあるすべてのオブジェクトを現在のディレクトリにダウンロードします。また、--exclude パラメータと --include パラメータを使用して、サフィックス *.log* が付いたオブジェクトのみをコピーします。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3 cp s3://example-s3-bucket1/logs/ . --recursive --exclude "*" --include "*.log"
```

詳細と例については、「AWS CLI コマンドリファレンス」の「cp」を参照してください。

AWS SDK の使用

AWS SDK を使用して Amazon S3 バケット内のすべてのオブジェクトをダウンロードする方法の例については、「Amazon Simple Storage Service (Amazon S3) バケットからすべてのオブジェクトを、ローカルディレクトリにダウンロードする」を参照してください。

さまざまな AWS SDK の使用に関する一般的な情報については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。

オブジェクトの一部のダウンロード

AWS CLI または REST API を使用してオブジェクトの一部をダウンロードできます。そのためには、追加のパラメータを使用して、オブジェクトのどの部分をダウンロードするかを指定します。

AWS CLI の使用

次のコマンド例は、*example-s3-bucket1* という名前のバケットの *folder/my_data* という名前のオブジェクトの特定のバイト範囲に対する GET リクエストを行います。リクエストでは、バイト範囲にプレフィックスとして bytes= を付ける必要があります。オブジェクト部分は、*my_data_range* という名前の出力ファイルにダウンロードされます。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3api get-object --bucket example-s3-bucket1 --key folder/my_data --range
 bytes=0-500 my_data_range
```

詳細と例については、「AWS CLI コマンドリファレンス」の「get-object」を参照してください。

HTTP Range ヘッダーの詳細については、RFC Editor ウェブサイトで「RFC 9110」を参照してください。

> **Note**
>
> Amazon S3 は、1 回の GET リクエストで複数範囲のデータを取得することはサポートしていません。

REST API の使用

Amazon S3 から複数のオブジェクト部分を取得するには、REST API で `partNumber` パラメータと `Range` パラメータを使用できます。詳細については、「Amazon Simple Storage Service API リファレンス」の「GetObject」を参照してください。

別の AWS アカウントに属するオブジェクトのダウンロード

署名付き URL を使用して、バケットポリシーを更新せずに、オブジェクトへの時間制限付きのアクセスを他のユーザーに許可できます。

署名付き URL をブラウザに入力するか、プログラムで使用してオブジェクトをダウンロードできます。URL で使用する認証情報は、URL を生成した AWS ユーザーの認証情報です。URL を作成すると、署名付き URL を持つすべてのユーザーが、URL の有効期限が切れるまで該当するオブジェクトをダウンロードできます。

S3 コンソールでの署名付き URL の使用

Amazon S3 コンソールで次の手順に従い、オブジェクトを共有するための署名付き URL を生成できます。コンソールを使用する場合、署名付き URL の最大有効期限は作成時点から 12 時間です。

Amazon S3 コンソールを使用して署名付き URL を生成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、署名付き URL を取得するオブジェクトが含まれているバケットの名前を選択します。
4. [オブジェクト] リストで、署名付き URL を作成するオブジェクトを選択します。
5. [オブジェクトアクション] メニューで、[署名付き URL で共有] を選択します。
6. 署名付き URL の有効期間を指定します。
7. [Create presigned URL] (署名付き URL を作成) を選択します。

8. 確認メッセージが表示されると、URL は自動的にクリップボードにコピーされます。署名済み URL を再度コピーする必要がある場合は、署名済み URL をコピーするボタンが表示されます。
9. オブジェクトをダウンロードするには、URL を任意のブラウザに貼り付けると、オブジェクトのダウンロードが試行されます。

署名付き URL とその作成方法の詳細については、「署名付き URL の使用」を参照してください。

アーカイブされたオブジェクトのダウンロード

アクセス頻度の低いオブジェクトのストレージコストを削減するには、それらのオブジェクトをアーカイブできます。オブジェクトをアーカイブすると、そのオブジェクトは低コストのストレージに移動されるため、リアルタイムでアクセスすることはできません。アーカイブされたオブジェクトをダウンロードするには、まず復元する必要があります。

ストレージクラスに応じて、アーカイブされたオブジェクトは、数分または数時間で復元できます。アーカイブされたオブジェクトは、Amazon S3 コンソール、S3 バッチオペレーション、Amazon S3 REST API、AWS SDK、および AWS Command Line Interface (AWS CLI) を使用して復元できます。

手順については、アーカイブされたオブジェクトの復元 を参照してください。アーカイブされたオブジェクトは、復元後にダウンロードできます。

オブジェクトのダウンロードに関するトラブルシューティング

Amazon S3 からオブジェクトをダウンロードする場合、アクセス許可が不十分だったり、バケットや AWS Identity and Access Management (IAM) ユーザーポリシーが正しくなかったりすると、エラーが発生する可能性があります。これらの問題により、アクセス拒否 (403 Forbidden) エラーが発生し、Amazon S3 がリソースへのアクセスを許可できない場合があります。

アクセス拒否 (403 Forbidden) エラーの一般的な原因については、「Amazon S3 でのアクセス拒否 (403 Forbidden) エラーのトラブルシューティング」を参照してください。

オブジェクトの整合性をチェックする

Amazon S3 は、チェックサム値を使用して、Amazon S3 にアップロードまたはダウンロードするデータの整合性を検証します。さらに、Amazon S3 に保存するオブジェクトについて、別のチェックサム値を計算するようにリクエストすることもできます。データをアップロードまたはコピーするときに使用するチェックサムアルゴリズムのいずれかを選択できます。Amazon S3 は、このアル

ゴリズムを使用して追加のチェックサム値を計算し、オブジェクトメタデータの一部として保存します。追加のチェックサムを使用してデータの整合性を検証する方法の詳細については、「Tutorial: Checking the integrity of data in Amazon S3 with additional checksums」(チュートリアル: Amazon S3 のデータの整合性を追加のチェックサムで確認する) を参照してください。

オブジェクトをアップロードするときに、オプションで事前計算されたチェックサムをリクエストの一部として含めることができます。Amazon S3 は、指定されたアルゴリズムを使用して計算したチェックサムと、指定されたチェックサムを比較します。2 つの値が一致しない場合、Amazon S3 はエラーを報告します。

サポートされているチェックサムアルゴリズムの使用

Amazon S3 では、アップロードまたはダウンロード中にデータを検証するために使用されるチェックサムアルゴリズムを選択するオプションがあります。次のセキュアハッシュアルゴリズム (SHA) または巡回冗長検査 (CRC) のデータ整合性チェックアルゴリズムのいずれかを選択できます。

- CRC32
- CRC32C
- SHA-1
- SHA-256

オブジェクトをアップロードするときに、使用するアルゴリズムを指定できます。

- AWS Management Console を使用するときには、使用するチェックサムのアルゴリズムを選択します。その場合、オプションでオブジェクトのチェックサム値を指定できます。Amazon S3 は、オブジェクトを受信すると、指定したアルゴリズムを使用してチェックサムを計算します。2 つの値が一致しない場合、Amazon S3 はエラーを生成します。
- SDK を使用しているときには、`x-amz-sdk-checksum-algorithm` パラメータの値を、Amazon S3 がチェックサムの計算時に 使用するアルゴリズムに設定できます。Amazon S3 はチェックサム値を自動的に計算します。
- REST API を使用しているときには、`x-amz-sdk-checksum-algorithm` パラメータを使用しません。代わりに、アルゴリズム固有のヘッダーのいずれかを使用します (例えば、`x-amz-checksum-crc32`)。

オブジェクトのアップロードの詳細については、「オブジェクトのアップロード」を参照してください。

Amazon S3 にすでにアップロードされているオブジェクトにこれらのチェックサム値を適用するには、オブジェクトをコピーします。オブジェクトをコピーするとき、既存のチェックサムアルゴリズムを使用するか、新しいチェックサムアルゴリズムを使用するかを指定できます。S3 バッチ操作など、サポートされているオブジェクトのコピーメカニズムを使用するときには、チェックサムアルゴリズムを指定できます。S3 バッチオペレーションについて詳しくは、Amazon S3 オブジェクトでの大規模なバッチ操作の実行 を参照してください。

> **Important**
>
> 追加のチェックサムを含むマルチパートアップロードを使用する場合、マルチパート番号には連続するパート番号を使用する必要があります。追加のチェックサムを使用するとき、連続しないパート番号でマルチパートアップロードリクエストを完了しようとすると、Amazon S3 は HTTP `500 Internal Server Error` エラーを生成します。

オブジェクトをアップロードした後、チェックサム値を取得し、同じアルゴリズムを使用して計算された事前計算済みまたは以前に保存されたチェックサム値と比較できます。

S3 コンソールの使用

コンソールの使用方法およびオブジェクトのアップロード時に使用するチェックサムアルゴリズムの指定方法の詳細については、「オブジェクトのアップロード」および「チュートリアル: チェックサムを追加して Amazon S3 のデータの整合性をチェックする」を参照してください。

AWS SDK の使用

次の例は、AWS SDK を使用して、マルチパートアップロードで大きなファイルをアップロードする方法、大きなファイルをダウンロードする方法、およびマルチパートアップロードファイルを検証する方法を示し、すべてファイル検証に SHA-256 を使用しています。

Java

Example 例: SHA-256 を使用して大きなファイルをアップロード、ダウンロード、および検証する

作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.core.ResponseInputStream;
```

```
    import software.amazon.awssdk.core.sync.RequestBody;
    import software.amazon.awssdk.regions.Region;
    import software.amazon.awssdk.services.s3.S3Client;
    import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
    import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
    import software.amazon.awssdk.services.s3.model.ChecksumMode;
    import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
    import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
    import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
    import software.amazon.awssdk.services.s3.model.CompletedPart;
    import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
    import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
    import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
    import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
    import software.amazon.awssdk.services.s3.model.GetObjectRequest;
    import software.amazon.awssdk.services.s3.model.GetObjectResponse;
    import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
    import software.amazon.awssdk.services.s3.model.ObjectAttributes;
    import software.amazon.awssdk.services.s3.model.PutObjectTaggingRequest;
    import software.amazon.awssdk.services.s3.model.Tag;
    import software.amazon.awssdk.services.s3.model.Tagging;
    import software.amazon.awssdk.services.s3.model.UploadPartRequest;
    import software.amazon.awssdk.services.s3.model.UploadPartResponse;

    import java.io.File;
    import java.io.FileInputStream;
    import java.io.FileOutputStream;
    import java.io.IOException;
    import java.io.InputStream;
    import java.io.OutputStream;
    import java.nio.ByteBuffer;
    import java.security.MessageDigest;
    import java.security.NoSuchAlgorithmException;
    import java.util.ArrayList;
    import java.util.Base64;
    import java.util.List;

    public class LargeObjectValidation {
        private static String FILE_NAME = "sample.file";
        private static String BUCKET = "sample-bucket";
        //Optional, if you want a method of storing the full multipart object
checksum in S3.
        private static String CHECKSUM_TAG_KEYNAME = "fullObjectChecksum";
```

```
        //If you have existing full-object checksums that you need to validate
 against, you can do the full object validation on a sequential upload.
        private static String SHA256_FILE_BYTES = "htCM5g7ZNdoSw8bN/
mkgiAhXt5MFoVowVg+LE9aIQmI=";
        //Example Chunk Size - this must be greater than or equal to 5MB.
        private static int CHUNK_SIZE = 5 * 1024 * 1024;

        public static void main(String[] args) {
            S3Client s3Client = S3Client.builder()
                    .region(Region.US_EAST_1)
                    .credentialsProvider(new AwsCredentialsProvider() {
                        @Override
                        public AwsCredentials resolveCredentials() {
                            return new AwsCredentials() {
                                @Override
                                public String accessKeyId() {
                                    return Constants.ACCESS_KEY;
                                }

                                @Override
                                public String secretAccessKey() {
                                    return Constants.SECRET;
                                }
                            };
                        }
                    })
                    .build();
            uploadLargeFileBracketedByChecksum(s3Client);
            downloadLargeFileBracketedByChecksum(s3Client);
            validateExistingFileAgainstS3Checksum(s3Client);
        }

        public static void uploadLargeFileBracketedByChecksum(S3Client s3Client) {
            System.out.println("Starting uploading file validation");
            File file = new File(FILE_NAME);
            try (InputStream in = new FileInputStream(file)) {
                MessageDigest sha256 = MessageDigest.getInstance("SHA-256");
                CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                        .bucket(BUCKET)
                        .key(FILE_NAME)
                        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
                        .build();
```

```
                CreateMultipartUploadResponse createdUpload =
s3Client.createMultipartUpload(createMultipartUploadRequest);
                List<CompletedPart> completedParts = new ArrayList<CompletedPart>();
                int partNumber = 1;
                byte[] buffer = new byte[CHUNK_SIZE];
                int read = in.read(buffer);
                while (read != -1) {
                    UploadPartRequest uploadPartRequest =
UploadPartRequest.builder()

.partNumber(partNumber).uploadId(createdUpload.uploadId()).key(FILE_NAME).bucket(BUCKET).ch
                    UploadPartResponse uploadedPart =
s3Client.uploadPart(uploadPartRequest,
RequestBody.fromByteBuffer(ByteBuffer.wrap(buffer, 0, read)));
                    CompletedPart part =
CompletedPart.builder().partNumber(partNumber).checksumSHA256(uploadedPart.checksumSHA256()
                    completedParts.add(part);
                    sha256.update(buffer, 0, read);
                    read = in.read(buffer);
                    partNumber++;
                }
                String fullObjectChecksum =
Base64.getEncoder().encodeToString(sha256.digest());
                if (!fullObjectChecksum.equals(SHA256_FILE_BYTES)) {
                    //Because the SHA256 is uploaded after the part is uploaded; the
upload is bracketed and the full object can be fully validated.

s3Client.abortMultipartUpload(AbortMultipartUploadRequest.builder().bucket(BUCKET).key(FILE
                    throw new IOException("Byte mismatch between stored checksum and
upload, do not proceed with upload and cleanup");
                }
                CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder().parts(completedParts).build();
                CompleteMultipartUploadResponse completedUploadResponse =
s3Client.completeMultipartUpload(

CompleteMultipartUploadRequest.builder().bucket(BUCKET).key(FILE_NAME).uploadId(createdUplo
                Tag checksumTag =
Tag.builder().key(CHECKSUM_TAG_KEYNAME).value(fullObjectChecksum).build();
                //Optionally, if you need the full object checksum stored with the
file; you could add it as a tag after completion.

s3Client.putObjectTagging(PutObjectTaggingRequest.builder().bucket(BUCKET).key(FILE_NAME).t
            } catch (IOException | NoSuchAlgorithmException e) {
```

```
                e.printStackTrace();
            }
            GetObjectAttributesResponse
                    objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                    .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
            System.out.println(objectAttributes.objectParts().parts());
            System.out.println(objectAttributes.checksum().checksumSHA256());
        }

        public static void downloadLargeFileBracketedByChecksum(S3Client s3Client) {
            System.out.println("Starting downloading file validation");
            File file = new File("DOWNLOADED_" + FILE_NAME);
            try (OutputStream out = new FileOutputStream(file)) {
                GetObjectAttributesResponse
                        objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                        .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
                //Optionally if you need the full object checksum, you can grab a
tag you added on the upload
                List<Tag> objectTags =
s3Client.getObjectTagging(GetObjectTaggingRequest.builder().bucket(BUCKET).key(FILE_NAME).b
                String fullObjectChecksum = null;
                for (Tag objectTag : objectTags) {
                    if (objectTag.key().equals(CHECKSUM_TAG_KEYNAME)) {
                        fullObjectChecksum = objectTag.value();
                        break;
                    }
                }
                MessageDigest sha256FullObject =
MessageDigest.getInstance("SHA-256");
                MessageDigest sha256ChecksumOfChecksums =
MessageDigest.getInstance("SHA-256");

                //If you retrieve the object in parts, and set the ChecksumMode to
enabled, the SDK will automatically validate the part checksum
                for (int partNumber = 1; partNumber <=
objectAttributes.objectParts().totalPartsCount(); partNumber++) {
                    MessageDigest sha256Part = MessageDigest.getInstance("SHA-256");
                    ResponseInputStream<GetObjectResponse> response =
s3Client.getObject(GetObjectRequest.builder().bucket(BUCKET).key(FILE_NAME).partNumber(part
                    GetObjectResponse getObjectResponse = response.response();
```

```
                    byte[] buffer = new byte[CHUNK_SIZE];
                    int read = response.read(buffer);
                    while (read != -1) {
                        out.write(buffer, 0, read);
                        sha256FullObject.update(buffer, 0, read);
                        sha256Part.update(buffer, 0, read);
                        read = response.read(buffer);
                    }
                    byte[] sha256PartBytes = sha256Part.digest();
                    sha256ChecksumOfChecksums.update(sha256PartBytes);
                    //Optionally, you can do an additional manual validation again
 the part checksum if needed in addition to the SDK check
                    String base64PartChecksum =
 Base64.getEncoder().encodeToString(sha256PartBytes);
                    String base64PartChecksumFromObjectAttributes =
 objectAttributes.objectParts().parts().get(partNumber - 1).checksumSHA256();
                    if (!
base64PartChecksum.equals(getObjectResponse.checksumSHA256()) || !
base64PartChecksum.equals(base64PartChecksumFromObjectAttributes)) {
                        throw new IOException("Part checksum didn't match for the
 part");
                    }
                    System.out.println(partNumber + " " + base64PartChecksum);
                }
                //Before finalizing, do the final checksum validation.
                String base64FullObject =
 Base64.getEncoder().encodeToString(sha256FullObject.digest());
                String base64ChecksumOfChecksums =
 Base64.getEncoder().encodeToString(sha256ChecksumOfChecksums.digest());
                if (fullObjectChecksum != null && !
fullObjectChecksum.equals(base64FullObject)) {
                    throw new IOException("Failed checksum validation for full
 object");
                }
                System.out.println(fullObjectChecksum);
                String base64ChecksumOfChecksumFromAttributes =
 objectAttributes.checksum().checksumSHA256();
                if (base64ChecksumOfChecksumFromAttributes != null && !
base64ChecksumOfChecksums.equals(base64ChecksumOfChecksumFromAttributes)) {
                    throw new IOException("Failed checksum validation for full
 object checksum of checksums");
                }
                System.out.println(base64ChecksumOfChecksumFromAttributes);
                out.flush();
```

```
            } catch (IOException | NoSuchAlgorithmException e) {
                //Cleanup bad file
                file.delete();
                e.printStackTrace();
            }
        }

        public static void validateExistingFileAgainstS3Checksum(S3Client s3Client)
{
            System.out.println("Starting existing file validation");
            File file = new File("DOWNLOADED_" + FILE_NAME);
            GetObjectAttributesResponse
                    objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                    .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
            try (InputStream in = new FileInputStream(file)) {
                MessageDigest sha256ChecksumOfChecksums =
MessageDigest.getInstance("SHA-256");
                MessageDigest sha256Part = MessageDigest.getInstance("SHA-256");
                byte[] buffer = new byte[CHUNK_SIZE];
                int currentPart = 0;
                int partBreak =
objectAttributes.objectParts().parts().get(currentPart).size();
                int totalRead = 0;
                int read = in.read(buffer);
                while (read != -1) {
                    totalRead += read;
                    if (totalRead >= partBreak) {
                        int difference = totalRead - partBreak;
                        byte[] partChecksum;
                        if (totalRead != partBreak) {
                            sha256Part.update(buffer, 0, read - difference);
                            partChecksum = sha256Part.digest();
                            sha256ChecksumOfChecksums.update(partChecksum);
                            sha256Part.reset();
                            sha256Part.update(buffer, read - difference,
difference);
                        } else {
                            sha256Part.update(buffer, 0, read);
                            partChecksum = sha256Part.digest();
                            sha256ChecksumOfChecksums.update(partChecksum);
                            sha256Part.reset();
                        }
```

```
                        String base64PartChecksum =
 Base64.getEncoder().encodeToString(partChecksum);
                        if (!
base64PartChecksum.equals(objectAttributes.objectParts().parts().get(currentPart).checksumSH
 {
                            throw new IOException("Part checksum didn't match S3");
                        }
                        currentPart++;
                        System.out.println(currentPart + " " + base64PartChecksum);
                        if (currentPart <
 objectAttributes.objectParts().totalPartsCount()) {
                            partBreak +=
 objectAttributes.objectParts().parts().get(currentPart - 1).size();
                        }
                    } else {
                        sha256Part.update(buffer, 0, read);
                    }
                    read = in.read(buffer);
                }
                if (currentPart != objectAttributes.objectParts().totalPartsCount())
 {
                    currentPart++;
                    byte[] partChecksum = sha256Part.digest();
                    sha256ChecksumOfChecksums.update(partChecksum);
                    String base64PartChecksum =
 Base64.getEncoder().encodeToString(partChecksum);
                    System.out.println(currentPart + " " + base64PartChecksum);
                }

                String base64CalculatedChecksumOfChecksums =
 Base64.getEncoder().encodeToString(sha256ChecksumOfChecksums.digest());
                System.out.println(base64CalculatedChecksumOfChecksums);
                System.out.println(objectAttributes.checksum().checksumSHA256());
                if (!
base64CalculatedChecksumOfChecksums.equals(objectAttributes.checksum().checksumSHA256()))
 {
                    throw new IOException("Full object checksum of checksums don't
 match S3");
                }

            } catch (IOException | NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
        }
```

```
    }
```

REST API の使用

REST リクエストを送信して、チェックサムの値を持つオブジェクトをアップロードし、PutObject でデータの整合性を検証できます。GetObject または HeadObject を使用して、オブジェクトのチェックサムの値を取得することもできます。

AWS CLI の使用

単一のオペレーションで、最大 5 GB のオブジェクトをアップロードする PUT リクエストを送信できます。詳細については、「AWS CLI コマンドリファレンス」の `PutObject` を参照してください。また、`get-object` と `head-object` を使用して、すでにアップロードされたオブジェクトのチェックサムを取得し、データの整合性を検証することもできます。

詳細については、「AWS Command Line Interface ユーザーガイド」の「Amazon S3 CLI のよくある質問」を参照してください。

オブジェクトをアップロードするときに Content-MD5 を使用する

アップロード後にオブジェクトの整合性を検証するもう 1 つの方法は、アップロード時にオブジェクトの MD5 ダイジェストを指定することです。オブジェクトの MD5 ダイジェストを計算した場合、`Content-MD5` ヘッダーを使用することで、PUT コマンドでダイジェストを指定できます。

オブジェクトをアップロードした後、Amazon S3 はオブジェクトの MD5 ダイジェストを計算し、指定した値と比較します。リクエストは、2 つのダイジェストが一致した場合にのみ成功します。

MD5 ダイジェストを指定する必要はありませんが、アップロードプロセスの一環としてオブジェクトの整合性を検証するために使用できます。

Content-MD5 と ETag を使用して、アップロードされたオブジェクトを検証する

オブジェクトのエンティティタグ (ETag) は、そのオブジェクトの特定のバージョンを表します。ETag は、オブジェクトのコンテンツに加えられた変更のみを反映し、メタデータに加えられた変更を反映しないことに注意してください。オブジェクトのメタデータのみが変更された場合、ETag は同じままです。

オブジェクトによっては、オブジェクトの ETag がオブジェクトデータの MD5 ダイジェストである場合があります。

- オブジェクトが PutObject、PostObject、または CopyObject オペレーションによって、または AWS Management Console を介して作成され、そのオブジェクトがプレーンテキストか、Amazon S3 マネージドキーを使用したサーバー側の暗号化 (SSE-S3) によって暗号化されている場合、そのオブジェクトの ETag は、オブジェクトデータの MD5 ダイジェストです。
- オブジェクトが PutObject、PostObject、または CopyObject オペレーションによって、または AWS Management Console を介して作成され、そのオブジェクトがお客様が用意したキーを使用したサーバー側の暗号化 (SSE-C) または AWS Key Management Service (AWS KMS) キーを使用したサーバー側の暗号化 (SSE-KMS) によって暗号化されている場合、そのオブジェクトの ETag は、オブジェクトデータの MD5 ダイジェストではありません。
- オブジェクトが Multipart Upload または Part Copy オペレーションによって作成された場合、暗号化の方法に関係なく、オブジェクトの ETag は MD5 ダイジェストではありません。オブジェクトが 16 MB より大きい場合、AWS Management Console はそのオブジェクトをマルチパートアップロードとしてアップロードまたはコピーするため、ETag は MD5 ダイジェストではありません。

ETag がオブジェクトの Content-MD5 ダイジェストであるオブジェクトの場合、オブジェクトの ETag 値を計算済みまたは以前に保存した Content-MD5 ダイジェストと比較できます。

追跡チェックサムの使用

Amazon S3 にオブジェクトをアップロードするときには、オブジェクトの事前計算されたチェックサムを指定するか、AWS SDK を使用して、追跡チェックサムを自動的に作成できます。追跡チェックサムを使用した場合、Amazon S3 は指定されたアルゴリズムを使用してチェックサムを自動的に生成し、それを使用してアップロード中にオブジェクトの整合性を検証します。

AWS SDK を使用しているときに追跡チェックサムを作成するには、ChecksumAlgorithm パラメータに任意のアルゴリズムを指定します。SDK は、そのアルゴリズムを使用してオブジェクト (またはオブジェクトパート) のチェックサムを計算し、アップロードリクエストの最後に自動的に追加します。この動作により、Amazon S3 はデータの検証とアップロードを単一のパスで実行するため、時間を節約できます。

> **Important**
>
> S3 オブジェクト Lambda を使用している場合、S3 オブジェクト Lambda へのすべてのリクエストは、s3 の代わりに s3-object-lambda を使用して署名されます。この動作は、追跡チェックサム値のシグネチャに影響します。S3 Object Lambda, の詳細については、「S3 Object Lambda を使用したオブジェクトの変換」を参照してください。

マルチパートアップロードにパートレベルのチェックサムを使用する

オブジェクトが Amazon S3 にアップロードされるときには、単一のオブジェクトとしてアップロードされるか、マルチパートアップロードプロセスを通じてアップロードされます。16 MB を超え、コンソールからアップロードされるオブジェクトは、マルチパートアップロードを使用して自動的にアップロードされます。マルチパートアップロードの詳細については、「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。

オブジェクトがマルチパートアップロードとしてアップロードされるとき、オブジェクトの ETag はオブジェクト全体の MD5 ダイジェストではありません。Amazon S3 は、アップロードされた個々のパートの MD5 ダイジェストを計算します。MD5 ダイジェストは、最終的なオブジェクトの ETag を決定するために使用されます。Amazon S3 は MD5 ダイジェストのバイトを連結し、これらの連結値の MD5 ダイジェストを計算します。ETag を作成するための最後のステップは、Amazon S3 がパートの総数を含むダッシュを最後に追加するときです。

例えば、ETag が `C9A5A6878D97B48CC965C1E41859F034-14` のマルチパートアップロードでアップロードされたオブジェクトを考えてみましょう。この場合、`C9A5A6878D97B48CC965C1E41859F034` は、連結されたすべてのダイジェストの MD5 ダイジェストです。`-14` は、このオブジェクトのマルチパートアップロードに 14 個のパートが関連付けられていることを示します。

マルチパートオブジェクトに対して追加のチェックサム値を有効にした場合、Amazon S3 は、指定されたチェックサムアルゴリズムを使用して、個々のパートのチェックサムを計算します。完了したオブジェクトのチェックサムは、Amazon S3 がマルチパートアップロードの MD5 ダイジェストを計算するのと同じ方法で計算されます。このチェックサムを使用して、オブジェクトの整合性を検証できます。

オブジェクト全体を構成するパートの数など、オブジェクトに関する情報を取得するには、GetObjectAttributes オペレーションを使用します。追加のチェックサムを使用すると、各パートのチェックサム値を含む個々のパートの情報を回復することもできます。

完了したアップロードでは、GetObject または HeadObject オペレーションを使用し、1 つのパートのパート番号またはバイト範囲を指定することによって、個々のパートのチェックサムを取得することもできます。まだ進行中のマルチパートアップロードの個々の部分のチェックサム値を取得する場合は、ListParts を使用します。

Amazon S3 はマルチパートオブジェクトのチェックサムを計算するので、オブジェクトをコピーした場合、チェックサム値が変更されることがあります。SDK または REST API を使用しており、CopyObject を呼び出した場合、Amazon S3 は `CopyObject` API オペレーションのサイズ制限

までオブジェクトをコピーします。Amazon S3 は、オブジェクトが単一のリクエストでアップロードされたか、マルチパートアップロードの一部としてアップロードされたかにかかわらず、このコピーを単一のアクションとして実行します。copy コマンドでは、オブジェクトのチェックサムは、完全なオブジェクトの直接チェックサムです。オブジェクトが最初にマルチパートアップロードを使用してアップロードされた場合、データにはない場合でも、チェックサム値が変更されます。

> **Note**
>
> `CopyObject` API オペレーションのサイズ制限よりも大きいオブジェクトは、マルチパート copy コマンドを使用する必要があります。

> **Important**
>
> AWS Management Console を使用していくつかの操作を実行するとき、オブジェクトのサイズが 16 MB より大きい場合、Amazon S3 はマルチパートアップロードを使用します。この場合、チェックサムは完全なオブジェクトの直接チェックサムではなく、個々のパートのチェックサム値に基づく計算です。
> 例えば、REST API を使用してシングルパートダイレクトアップロードとしてアップロードした 100 MB のサイズのオブジェクトがあるとします。この場合のチェックサムは、オブジェクト全体のチェックサムです。後でコンソールを使用してオブジェクトの名前変更、コピー、ストレージクラスの変更、またはメタデータの編集を行った場合、Amazon S3 はマルチパートアップロード機能を使用してオブジェクトを更新します。その結果、Amazon S3 は、個々のパートのチェックサム値に基づいて計算されるオブジェクトの新しいチェックサム値を作成します。
> 前述のコンソール操作のリストは、AWS Management Console で実行できるすべてのアクションの完全なリストではなく、その結果、Amazon S3 はマルチパートアップロード機能を使用してオブジェクトを更新します。コンソールを使用してサイズが 16 MB を超えるオブジェクトを操作する場合、チェックサム値はオブジェクト全体のチェックサムではない場合があることに注意してください。

Amazon S3 オブジェクトの削除

Amazon S3 コンソール、AWS SDK、AWS Command Line Interface (AWS CLI)、または REST API を使用して、Amazon S3 から 1 つ以上のオブジェクトを直接削除できます。S3 バケット内のすべてのオブジェクトによりストレージコストが生じるため、不要になったオブジェクトを削除する必要

があります。例えば、ログファイルを収集している場合は、不要になったファイルを削除することをお勧めします。ログファイルなどのオブジェクトを自動的に削除するライフサイクルルールをセットアップできます。詳細については、「the section called “ライフサイクル設定の指定”」を参照してください。

Amazon S3 の機能と料金の詳細については、「Amazon S3 の料金」を参照してください。

オブジェクトを削除するときは、以下の API オプションを利用できます。

- 単一のオブジェクトの削除 - Amazon S3 では DELETE (`DeleteObject`) API オペレーションを提供しています。これを使用すると、単一の HTTP リクエストで 1 つのオブジェクトを削除できます。
- 複数のオブジェクトの削除 - Amazon S3 では Multi-Object Delete (`DeleteObjects`) API を提供しています。これを使用すると、単一の HTTP リクエストで最大 1,000 個のオブジェクトを削除できます。

バージョニングが有効でないバケットからオブジェクトを削除する場合、オブジェクトキー名のみを指定します。ただし、バージョニングが有効なバケットからオブジェクトを削除する場合は、必要に応じてオブジェクトのバージョン ID を指定して、特定のバージョンのオブジェクトを削除することができます。

バージョニングが有効なバケットから、プログラムによってオブジェクトを削除する

バージョニングが有効なバケットの場合、複数のバージョンのオブジェクトがバケット内に存在する可能性があります。バージョニング対応のバケットを操作する場合は、delete API オペレーションで以下のオプションが可能です。

- バージョンを指定しない削除リクエストの指定 - オブジェクトのキーのみを指定し、バージョン ID は指定しません。この場合、Amazon S3 は削除マーカーを作成し、レスポンスでバージョン ID を返します。オブジェクトはバケットから消去されます。オブジェクトのバージョニングと削除マーカーの概念については、S3 バケットでのバージョニングの使用 を参照してください。
- バージョンを指定した削除リクエストの指定 - キーとバージョン ID の両方を指定します。この場合、以下の 2 通りの結果があり得ます。
  - バージョン ID が特定のオブジェクトバージョンにマッピングされている場合、Amazon S3 はその特定のバージョンのオブジェクトを削除します。

- バージョン ID がそのオブジェクトの削除マーカーに対応する場合、Amazon S3 は削除マーカーを削除します。オブジェクトはバケットに再び表示されます。

MFA 対応のバケットからのオブジェクトの削除

多工レメント認証 (MFA) 対応のバケットからオブジェクトを削除する場合は、以下の点に注意してください。

- 無効な MFA トークンを指定した場合、リクエストは常に失敗します。
- MFA 対応のバケットがあり、バージョンを指定した削除リクエストを行う (オブジェクトのキーとバージョン ID を指定する) 場合、有効な MFA トークンを指定しないとリクエストは失敗します。さらに、MFA 対応バケットに対して Multi-Object Delete API オペレーションを使用する場合、いずれかの削除がバージョンを指定した削除リクエストである場合 (つまり、オブジェクトのキーとバージョン ID を指定した場合)、MFA トークンを指定しないとリクエスト全体が失敗します。

ただし、以下の場合、リクエストは成功します。

- MFA 対応バケットを使用していて、バージョンを指定しない削除リクエストを行う (バージョニングされたオブジェクトを削除しない) 場合、MFA トークンを指定しなくても削除は成功します。
- Multi-Object Delete リクエストで、MFA 対応バケットからバージョニング非対応のオブジェクトのみを削除するように指定した場合、MFA トークンを指定しなくても削除は成功します。

MFA 削除の詳細については、「MFA 削除の設定」を参照してください。

トピック

- 単一のオブジェクトの削除
- 複数のオブジェクトの削除

単一のオブジェクトの削除

Amazon S3 コンソールまたは DELETE API を使用して、S3 バケットから 1 つの既存のオブジェクトを削除できます。Amazon S3 でのオブジェクト削除の詳細については、「Amazon S3 オブジェクトの削除」を参照してください。

S3 バケット内のすべてのオブジェクトによりストレージコストが生じるため、不要になったオブジェクトを削除する必要があります。例えば、ログファイルを収集している場合は、不要になった

ファイルを削除することをお勧めします。ログファイルなどのオブジェクトを自動的に削除するライフサイクルルールをセットアップできます。詳細については、「the section called “ライフサイクル設定の指定”」を参照してください。

Amazon S3 の機能と料金の詳細については、「Amazon S3 の料金」を参照してください。

S3 コンソールの使用

Amazon S3 コンソールを使用してバケットから 1 つのオブジェクトを削除するには、以下の手順に従ってください。

> **Warning**
>
> Amazon S3 コンソールでオブジェクトまたは指定されたオブジェクトバージョンを完全に削除した場合、削除を元に戻すことはできません。

バージョニングが有効化または停止されているオブジェクトを削除するには

> **Note**
>
> バージョニングが停止されたバケット内のオブジェクトのバージョン ID が NULL とマークされている場合、S3 は以前のバージョンが存在しないためオブジェクトを完全に削除します。ただし、バージョニングが停止されたバケット内のオブジェクトに有効なバージョン ID が一覧表示されている場合、S3 は削除されたオブジェクトの削除マーカーを作成し、オブジェクトの以前のバージョンを保持します。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、オブジェクトを削除するバケットの名前を選択します。
3. オブジェクトを選択した後、[削除] を選択します。
4. [指定されたオブジェクト] に表示されているオブジェクトが削除されていることを確認するには、[オブジェクトを削除しますか?] テキスト ボックスに「**`delete`**」と入力します。

バージョニングが有効なバケット内の特定のオブジェクトバージョンを完全に削除するには

> **Warning**
>
> Amazon S3 で指定されたオブジェクトバージョンを完全に削除した場合、削除を元に戻すことはできません。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、オブジェクトを削除するバケットの名前を選択します。
3. 削除するオブジェクトを選択します。
4. [バージョンを表示] トグルを選択します。
5. オブジェクトバージョンを選択した後、[削除] を選択します。
6. [指定されたオブジェクト] に表示されている特定のオブジェクトバージョンが完全に削除されていることを確認するには、[オブジェクトを削除しますか?] テキスト ボックスに 「完全に削除」と入力します。Amazon S3 がその特定のオブジェクトバージョンを完全に削除します。

バージョニングが有効になっていない Amazon S3 バケット内のオブジェクトを完全に削除するには

> **Warning**
>
> Amazon S3 でオブジェクトを完全に削除した場合、削除を元に戻すことはできません。また、バージョニングが有効になっていないバケットの場合、削除は恒久的です。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、オブジェクトを削除するバケットの名前を選択します。
3. オブジェクトを選択した後、[削除] を選択します。
4. [指定されたオブジェクト] に表示されているオブジェクトが完全に削除されていることを確認するには、[オブジェクトを削除しますか?] テキスト ボックスに [完全に削除] と入力します。

> **Note**
>
> オブジェクトの削除に問題がある場合は、「バージョニングされたオブジェクトを完全に削除する」を参照してください。

AWS SDK の使用

以下の例では、AWS SDK を使用してバケットからオブジェクトを削除する方法を示します。詳細については、Amazon Simple Storage Service API リファレンスの DELETE Object を参照してください。

バケットで S3 バージョニングが有効化されている場合は、以下のオプションを使用できます。

- バージョン ID を指定して、特定のオブジェクトバージョンを削除します。
- バージョン ID を指定しないでオブジェクトを削除します。この場合、Amazon S3 によってオブジェクトに削除マーカーが追加されます。

S3 バージョニングの詳細については、S3 バケットでのバージョニングの使用 を参照してください。

Java

Example 例 1: オブジェクトを削除する (バージョニング非対応のバケット)

この例では、バケットでバージョン管理が有効になっておらず、オブジェクトにバージョン ID がないことを前提としてます。削除リクエストで、オブジェクトキーのみを指定し、バージョン ID を指定しません。

作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectRequest;
```

```
import java.io.IOException;

public class DeleteObjectNonVersionedBucket {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            s3Client.deleteObject(new DeleteObjectRequest(bucketName, keyName));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example 例 2: オブジェクトを削除する (バージョニング対応のバケット)

次の例では、バージョニング対応のバケットからオブジェクトを削除します。オブジェクトのキー名とバージョン ID を指定して特定のオブジェクトを削除します。

この例では、次のような処理を実行します。

1. サンプルオブジェクトをバケットに追加します。Amazon S3 は新しく追加されたオブジェクトのバージョン ID を返します。この例では、このバージョン ID を削除リクエストで使用します。
2. オブジェクトのキー名とバージョン ID の両方を指定してオブジェクトバージョンを削除します。オブジェクトに他のバージョンがない場合、Amazon S3 はオブジェクトを完全に削除します。それ以外の場合、Amazon S3 は指定されたバージョンのみを削除します。

> **Note**
>
> オブジェクトのバージョン ID は、ListVersions リクエストを送信して取得できます。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.DeleteVersionRequest;
import com.amazonaws.services.s3.model.PutObjectResult;

import java.io.IOException;

public class DeleteObjectVersionEnabledBucket {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Check to ensure that the bucket is versioning-enabled.
            String bucketVersionStatus =
 s3Client.getBucketVersioningConfiguration(bucketName).getStatus();
            if (!bucketVersionStatus.equals(BucketVersioningConfiguration.ENABLED))
 {
                System.out.printf("Bucket %s is not versioning-enabled.",
 bucketName);
            } else {
                // Add an object.
```

```
                PutObjectResult putResult = s3Client.putObject(bucketName, keyName,
                        "Sample content for deletion example.");
                System.out.printf("Object %s added to bucket %s\n", keyName,
 bucketName);

                // Delete the version of the object that we just created.
                System.out.println("Deleting versioned object " + keyName);
                s3Client.deleteVersion(new DeleteVersionRequest(bucketName, keyName,
 putResult.getVersionId()));
                System.out.printf("Object %s, version %s deleted\n", keyName,
 putResult.getVersionId());
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

以下の例では、バージョニング対応および非対応のバケットからオブジェクトを削除する方法を示します。S3 バージョニングの詳細については、S3 バケットでのバージョニングの使用 を参照してください。

Example バージョニング非対応バケットからのオブジェクトの削除

次の C# の例では、バージョニング非対応のバケットからオブジェクトを削除します。この例では、オブジェクトにバージョン ID がないため、バージョン ID を指定しないものとします。オブジェクトキーのみ指定します。

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
```

```
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DeleteObjectNonVersionedBucketTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            DeleteObjectNonVersionedBucketAsync().Wait();
        }
        private static async Task DeleteObjectNonVersionedBucketAsync()
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(deleteObjectRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
        }
    }
```

```
}
```

Example バージョニング対応バケットからのオブジェクトの削除

次の C# の例では、バージョニング対応のバケットからオブジェクトを削除します。オブジェクトのキー名とバージョン ID を指定してオブジェクトの特定のバージョンを削除します。

このコードでは、以下のタスクを実行します。

1. 指定したバケットの S3 バージョニングを有効にします (S3 バージョニングが既に有効になっている場合は効果がありません)。
2. サンプルオブジェクトをバケットに追加します。レスポンスとして、Amazon S3 は新しく追加されたオブジェクトのバージョン ID を返します。この例では、このバージョン ID を削除リクエストで使用します。
3. オブジェクトのキー名とバージョン ID の両方を指定してサンプルオブジェクトを削除します。

> **Note**
>
> オブジェクトのバージョン ID は、ListVersions リクエストを送信して取得することもできます。
>
> ```
> var listResponse = client.ListVersions(new ListVersionsRequest { BucketName
>  = bucketName, Prefix = keyName });
> ```

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DeleteObjectVersion
    {
        private const string bucketName = "*** versioning-enabled bucket name ***";
```

```
        private const string keyName = "*** Object Key Name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            CreateAndDeleteObjectVersionAsync().Wait();
        }

        private static async Task CreateAndDeleteObjectVersionAsync()
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(keyName);

                // Delete the object by specifying an object key and a version ID.
                DeleteObjectRequest request = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID
                };
                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
deleting an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
deleting an object", e.Message);
            }
        }

        static async Task<string> PutAnObject(string objectKey)
        {
            PutObjectRequest request = new PutObjectRequest
```

```
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!"
            };
            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
}
```

PHP

このトピックでは、バージョン 3 の AWS SDK for PHP のクラスを使用して、バージョニング非対応のバケットからオブジェクトを削除する方法を示します。バージョニング対応のバケットから 1 つのオブジェクトを削除する方法については、REST API の使用 を参照してください。

AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

次の PHP の例では、バケットからオブジェクトを削除します。この例で示す方法では、バージョニング非対応のバケットからオブジェクトを削除するため、削除リクエストではバケット名とオブジェクトキーのみを指定し、バージョン ID は指定しません。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// 1. Delete the object from the bucket.
try
{
    echo 'Attempting to delete ' . $keyname . '...' . PHP_EOL;
```

```
    $result = $s3->deleteObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);

    if ($result['DeleteMarker'])
    {
        echo $keyname . ' was deleted or does not exist.' . PHP_EOL;
    } else {
        exit('Error: ' . $keyname . ' was not deleted.' . PHP_EOL);
    }
}
catch (S3Exception $e) {
    exit('Error: ' . $e->getAwsErrorMessage() . PHP_EOL);
}

// 2. Check to see if the object was deleted.
try
{
    echo 'Checking to see if ' . $keyname . ' still exists...' . PHP_EOL;

    $result = $s3->getObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);

    echo 'Error: ' . $keyname . ' still exists.';
}
catch (S3Exception $e) {
    exit($e->getAwsErrorMessage());
}
```

Javascript

```
import { DeleteObjectCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js" // Helper function that creates Amazon
 S3 service client module.

export const bucketParams = { Bucket: "BUCKET_NAME", Key: "KEY" };

export const run = async () => {
  try {
```

```
    const data = await s3Client.send(new DeleteObjectCommand(bucketParams));
    console.log("Success. Object deleted.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

AWS CLI の使用

1 件のリクエストで 1 つのオブジェクトを削除するには、DELETE API を使用します。詳細については、DELETE オブジェクトを参照してください。CLI を使用してオブジェクトを削除する方法の詳細については、delete-object」参照してください。

REST API の使用

AWS SDK を使用して、1 つのオブジェクトを削除できます。ただし、アプリケーションで必要な場合は、REST リクエストを直接送信できます。詳細については、Amazon Simple Storage Service API リファレンスの DELETE Object を参照してください。

複数のオブジェクトの削除

S3 バケット内のすべてのオブジェクトによりストレージコストが生じるため、不要になったオブジェクトを削除する必要があります。例えば、ログファイルを収集している場合は、不要になったファイルを削除することをお勧めします。ログファイルなどのオブジェクトを自動的に削除するライフサイクルルールをセットアップできます。詳細については、「the section called “ライフサイクル設定の指定”」を参照してください。

Amazon S3 の機能と料金の詳細については、「Amazon S3 の料金」を参照してください。

Amazon S3 コンソール、AWS SDK、または REST API を使用して、S3 バケットから複数のオブジェクトを同時に削除できます。

S3 コンソールの使用

Amazon S3 コンソールを使用してバケットから複数のオブジェクトを削除するには、以下の手順に従ってください。

Warning

- 指定されたオブジェクトを削除すると元に戻せません。
- このアクションは、指定されたすべてのオブジェクトを削除します。フォルダを削除する場合は、削除アクションが完了するのを待ってから、フォルダに新しいオブジェクトを追加します。そうしなければ、新しいオブジェクトも削除される可能性があります。
- バージョニングが有効でないバケットのオブジェクトを削除すると、Amazon S3 がオブジェクトを恒久的に削除します。
- バケットバージョニングが有効または停止のバケットのオブジェクトを削除すると、Amazon S3 が削除マーカーを作成します。詳細については、「削除マーカーの使用」を参照してください。

バージョニングが有効化または停止されているオブジェクトを削除するには

Note

バージョニングが停止されたバケット内のオブジェクトのバージョン ID が NULL とマークされている場合、S3 は以前のバージョンが存在しないためオブジェクトを完全に削除します。ただし、バージョニングが停止されたバケット内のオブジェクトに有効なバージョン ID が一覧表示されている場合、S3 は削除されたオブジェクトの削除マーカーを作成し、オブジェクトの以前のバージョンを保持します。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット名] リストで、オブジェクトを削除するバケットの名前を選択します。
3. オブジェクトを選択した後、[削除] を選択します。
4. [指定されたオブジェクト] に表示されているオブジェクトが削除されていることを確認するには、[オブジェクトを削除しますか?] テキスト ボックスに「**delete**」と入力します。

バージョニングが有効なバケット内の特定のオブジェクトバージョンを完全に削除するには

> **Warning**
>
> Amazon S3 で指定されたオブジェクトバージョンを完全に削除した場合、削除を元に戻すことはできません。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット名] リストで、オブジェクトを削除するバケットの名前を選択します。
3. 削除したい オブジェクトを選択します。
4. [バージョンを表示] トグルを選択します。
5. オブジェクトバージョンを選択した後、[削除] を選択します。
6. [指定されたオブジェクト] に表示されている特定のオブジェクトバージョンが完全に削除されていることを確認するには、[オブジェクトを削除しますか?] テキスト ボックスに 「完全に削除」と入力します。Amazon S3 がその特定のオブジェクトバージョンを完全に削除します。

バージョニングが有効になっていない Amazon S3 バケット内のオブジェクトを完全に削除するには

> **Warning**
>
> Amazon S3 でオブジェクトを完全に削除した場合、削除を元に戻すことはできません。また、バージョニングが有効になっていないバケットの場合、削除は恒久的です。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット名] リストで、オブジェクトを削除するバケットの名前を選択します。
3. オブジェクトを選択した後、[削除] を選択します。
4. [指定されたオブジェクト] に表示されているオブジェクトが完全に削除されていることを確認するには、[オブジェクトを削除しますか?] テキスト ボックスに [完全に削除] と入力します。

> **Note**
>
> オブジェクトの削除に問題がある場合は、「バージョニングされたオブジェクトを完全に削除する」を参照してください。

AWS SDK の使用

AWS SDK を使用して複数のオブジェクトを削除する方法の例については、「AWS SDK または CLI で DeleteObjects を使用する」を参照してください。

さまざまな AWS SDK の使用に関する一般的な情報については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。

REST API の使用

AWS SDK を使用して、Multi-Object Delete API で複数のオブジェクトを削除できます。ただし、アプリケーションで必要な場合は、REST リクエストを直接送信できます。

詳細については、Amazon Simple Storage Service API リファレンスの複数オブジェクトの削除を参照してください。

オブジェクトの整理、リスト化、使用

Amazon S3 では、プレフィックスを使用してストレージを整理できます。プレフィックスは、バケット内のオブジェクトの論理グループです。プレフィックス値は、バケット内の同じディレクトリに同様のデータを保存するためのディレクトリ名に似ています。プログラムでオブジェクトをアップロードする場合は、プレフィックスを使用してデータを整理できます。

Amazon S3 コンソールでは、プレフィックスはフォルダと呼ばれます。バケットに移動すると、S3 コンソールですべてのオブジェクトとフォルダを表示できます。また、オブジェクトのプロパティなど、各オブジェクトに関する情報を表示することもできます。

Amazon S3 でのデータのリスト化と整理の詳細については、以下のトピックを参照してください。

トピック

- プレフィックスを使用してオブジェクトを整理する
- プログラムによるオブジェクトキーのリスト化
- フォルダを使用して Amazon S3 コンソールのオブジェクトを整理する

- Amazon S3 コンソールでのオブジェクトの概要の表示
- Amazon S3 コンソールでのオブジェクトのプロパティの表示

プレフィックスを使用してオブジェクトを整理する

プレフィックスを使用して、Amazon S3 バケットに保存するデータを整理できます。プレフィックスは、オブジェクトキー名の先頭にある文字列です。プレフィックスには、オブジェクトキー名の最大長 (1,024 バイト) を条件として、任意の長さを指定できます。プレフィックスは、ディレクトリと同様の方法でデータを整理する方法と考えることができます。ただし、プレフィックスはディレクトリではありません。

プレフィックスで検索すると、指定されたプレフィックスで始まるキーだけに結果が限定されます。区切り記号を使用すると、リストオペレーションによって、共通のプレフィックスを共有するすべてのキーが単一の要約リスト結果にまとめられます。

プレフィックスと区切り記号のパラメータの目的は、キーを階層的に構成および参照できるようにすることです。そのためには、まずバケットで使用する区切り記号を選択します。スラッシュ (/) など、キーの名前には使われないような記号を使用します。別の文字を区切り文字として使用できます。スラッシュ (/) 文字に特徴はありませんが、非常に一般的なプレフィックスの区切り文字です。次に、階層に含まれるすべてのレベルを連結し、各レベルを区切り記号で区切ることで、キーの名前を構成します。

例えば、都市に関する情報を格納するとしたら、大陸、国、州または県の順に情報を整理するのが自然でしょう。通常、これらの名前には句読点が含まれないため、区切り記号としてスラッシュ (/) を使用できます。以下の例ではスラッシュ (/) 区切り記号を使用しています。

- Europe/France/Nouvelle-Aquitaine/Bordeaux
- North America/Canada/Quebec/Montreal
- 北米/米国/ワシントン州/ベルビュー
- 北米/米国/ワシントン州/シアトル

世界中の全都市のデータをこの方法で格納した場合、階層のないフラットなキーネームスペースは管理しにくくなります。リストオペレーションで `Prefix` と `Delimiter` を使用することで、データをリストするために作成した階層を利用できます。例えば、米国のすべての州をリストするには、`Delimiter='/'` および `Prefix='North America/USA/'` をセットします。データが存在するカナダのすべての州をリストするには、`Delimiter='/'` および `Prefix='North America/Canada/'` をセットします。

区切り文字、プレフィックス、ネストフォルダの詳細については、「プレフィックスとネストされたフォルダの違い」を参照してください。

プレフィックスと区切り記号を使用したオブジェクトのリスト化

区切り記号付きのリストリクエストを発行すると、1 レベルの階層だけを参照でき、それより深いレベルで入れ子にされている (場合によっては数百万もの) キーはスキップおよび集約されます。例えば、次のキーを持つバケット (*DOC-EXAMPLE-BUCKET*) を使用する場合を考えます。

```
sample.jpg
```

```
photos/2006/January/sample.jpg
```

```
photos/2006/February/sample2.jpg
```

```
photos/2006/February/sample3.jpg
```

```
photos/2006/February/sample4.jpg
```

このサンプルバケットはルートレベルに sample.jpg オブジェクトだけを持ちます。バケット内のルートレベルオブジェクトだけを一覧表示するには、バケットに対する GET リクエストをスラッシュ (/) の区切り文字と共に送信します。Simple Storage Service (Amazon S3) からのレスポンスでは sample.jpg オブジェクトキーが返されます。このキーには / の区切り文字が含まれないためです。その他すべてのキーには区切り文字が含まれます。Amazon S3 によってこれらのキーがグループ化され、特定のプレフィックス値を持つ 1 つの CommonPrefixes 要素が返されます。この場合のプレフィックス値は photos/ であり、これはキーの先頭から、指定した区切り文字の最初の出現までのサブ文字列です。

Example

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>DOC-EXAMPLE-BUCKET</Name>
  <Prefix></Prefix>
  <Marker></Marker>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>sample.jpg</Key>
    <LastModified>2011-07-24T19:39:30.000Z</LastModified>
    <ETag>&quot;d1a7fb5eab1c16cb4f7cf341cf188c3d&quot;</ETag>
    <Size>6</Size>
    <Owner>
```

```
      <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
      <DisplayName>displayname</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <CommonPrefixes>
    <Prefix>photos/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

プログラムによるオブジェクトキーのリスト化の詳細については、「プログラムによるオブジェクトキーのリスト化」を参照してください 。

プログラムによるオブジェクトキーのリスト化

Amazon S3 では、キーはプレフィックス別に一覧表示できます。関連するキーの名前に共通のプレフィックスを選択し、階層を区切る特殊文字でこれらのキーをマークすることができます。その後、リストオペレーションを使用して、キーを階層的に選択および参照できます。これは、ファイルシステムにおいてディレクトリ内にファイルを格納するしくみに似ています。

Amazon S3 では、バケット内のキーを列挙するためのリストオペレーションを公開します。キーはバケット別またはプレフィックス別にリストされます。例えば、すべての英単語のキーを含む「dictionary」という名前のバケットがあるとします。そして、そのバケット内の文字「q」で始まるキーをすべてリストするための呼び出しを行うとします。リストの結果は常に UTF-8 バイナリ順で返されます。

SOAP および REST のリストオペレーションではいずれも、条件に一致するキーの名前と、各キーによって識別されるオブジェクトに関する情報を含む XML ドキュメントが返されます。

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

特殊な区切り記号で終わるプレフィックスを共有するキーのグループは、リストの目的で、その共通のプレフィックスによってロールアップできます。このような機能によりアプリケーションでは、キーを階層的に構成および参照できるようになります。ファイルシステムにおいてファイルをディレクトリに分けて整理するのに似ています。

例えば、英語以外の単語も含まれるように dictionary バケットを拡張するには、各単語にその言語と区切り記号のプレフィックスを付けることでキーを形成できます (「French/logical」など)。この命名スキームと階層リスト機能を使用すれば、フランス語の単語のリストだけを取得することもできます。また、対応している言語の最上位レベルのリストを参照することで、それより下の階層間に介在するすべてのキーを反復処理する必要もなくなります。リスト化における階層の役割については、プレフィックスを使用してオブジェクトを整理する を参照してください。

REST API

アプリケーションで必要な場合は、REST リクエストを直接送信できます。GET リクエストを送信して、バケット内の一部またはすべてのオブジェクトを返すことができます。または選択条件を使用して、バケット内のオブジェクトのサブセットを返すことができます。詳細については、Amazon Simple Storage Service API リファレンスの GET バケット (オブジェクトのリスト化) バージョン 2 を参照してください。

リスト実装の効率性

リスト化のパフォーマンスは、バケット内のキーの総数によって大きく影響されることはありません。`prefix`、`marker`、`maxkeys` または `delimiter` 引数の有無によって影響されることもありません。

複数ページの結果に対する反復処理

バケットに入れることのできるキーの数は実質的に無制限であるため、リストのクエリによっては結果が膨大な量になる可能性があります。大規模な結果セットを管理するため、Amazon S3 API ではページ分割をサポートして結果セットを複数のレスポンスに分割します。キーリストのレスポンスごとに最大 1,000 個のキーを含むページと、レスポンスが切り捨てられているかどうかを示すインジケータが返されます。すべてのキーを受信するまで、一連のキーリストリクエストを送信します。AWSこのページ分割は、SDK ラッパーライブラリでも行うことができます。

例

次のサンプルコードは、`ListObjects` を使用する方法を説明しています。

CLI

AWS CLI

次の例は、`list-objects` コマンドを使用して、指定されたバケット内のすべてのオブジェクトの名前を表示します。

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size:
 Size}'
```

この例では、--query 引数を使用して list-objects の出力を各オブジェクトのキー値とサイズでフィルタリングしています。

オブジェクトの詳細については、「Amazon S3 デベロッパーガイド」の「Working with Amazon S3 Objects」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「ListObjects」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、バケット「test-files」内のすべての項目に関する情報を取得します。

```
Get-S3Object -BucketName test-files
```

例 2: このコマンドは、バケット「test-files」内の項目「sample.txt」に関する情報を取得します。

```
Get-S3Object -BucketName test-files -Key sample.txt
```

例 3: このコマンドは、バケット「test-files」からプレフィックス「sample」を持つすべての項目に関する情報を取得します。

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference」の「ListObjects」を参照してください。

フォルダを使用して Amazon S3 コンソールのオブジェクトを整理する

Amazon S3 では、バケットとオブジェクトが主要なリソースであり、オブジェクトはバケットに格納されます。Amazon S3 はフラットな構造であり、ファイルシステムに見られる階層はありません。ただし、構造を分かりやすくするため、Amazon S3 コンソールでは、オブジェクトのグループ化の方法としてフォルダの概念をサポートしています。コンソールでは、これを行うために、グルー

プ化したオブジェクトに共有名のプレフィックスを使用します。つまり、グループ化したオブジェクトに共通の文字列で始まる名前が付けられます。この共通の文字列 (共有プレフィックス) がフォルダ名です。オブジェクト名はキー名とも呼ばれます。

例えば、コンソールに photos という名前のフォルダを作成し、その中に myphoto.jpg という名前のオブジェクトを保存できます。このオブジェクトはキー名 photos/myphoto.jpg で保存され、photos/ がプレフィックスになります。

ここでは、さらに 2 つの例を示します。

- バケットに logs/date1.txt、logs/date2.txt、logs/date3.txt という 3 つのオブジェクトがある場合、コンソールには logs という名前のフォルダが表示されます。コンソールでフォルダを開くと、date1.txt、date2.txt、date3.txt という 3 つのオブジェクトが表示されます。
- photos/2017/example.jpg という名前のオブジェクトがある場合、コンソールには、フォルダ photos を含む 2017 という名前のフォルダが表示されます。フォルダ 2017 にはオブジェクト example.jpg が含まれます。

フォルダ内にフォルダを作成できますが、バケット内にバケットを作成することはできません。オブジェクトをフォルダに直接アップロードしてコピーできます。フォルダは、作成、削除、公開することができますが、名前を変更することはできません。オブジェクトはフォルダ間でコピーすることができます。

> **Important**
>
> Amazon S3 にフォルダを作成すると、S3 は、指定したフォルダ名に設定されたキーを持つ 0 バイトのオブジェクトを作成します。例えば、バケットに photos という名前のフォルダを作成した場合、Amazon S3 コンソールは photos/ キーを使用して 0 バイトのオブジェクトを作成します。コンソールは、フォルダの考え方をサポートするために、このオブジェクトを作成します。
>
> Amazon S3 コンソールでは、キー名の最後 (末尾) の文字がスラッシュ (/) になっているすべてのオブジェクト (examplekeyname/ など) がフォルダとして扱われます。キー名の末尾の文字が / のオブジェクトは、Amazon S3 コンソールを使用してアップロードすることができません。ただし、Amazon S3 API で名前の末尾が / のオブジェクトは、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用してアップロードできます。
>
> 名前の末尾が / のオブジェクトは、Amazon S3 コンソールではフォルダとして扱われます。Amazon S3 コンソールには、このようなオブジェクトのコンテンツやメタデータが表示

されません。コンソールを使用して名前の末尾が / のオブジェクトをコピーすると、コピー先の場所に新しいフォルダが作成されますが、オブジェクトのデータとメタデータはコピーされません。

トピック

フォルダの作成

このセクションでは、Amazon S3 コンソールを使用してフォルダを作成する方法について説明します。

> **Important**
>
> バケットポリシーが原因で、このバケットへのオブジェクトのアップロードを、タグ、メタデータ、またはアクセスコントロールリスト (ACL) の被付与者なしで行うことができない場合は、次の手順を使用してフォルダを作成することはできません。代わりに、空のフォルダをアップロードし、以下の設定をアップロード設定で指定します。

フォルダを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、フォルダを作成するバケットの名前を選択します。
4. バケットポリシーが原因で、暗号化をせずにこのバケットへのオブジェクトのアップロードができない場合は、[Server-side encryption] (サーバー側の暗号化) で [Enable] を選択する必要があります。
5. [Create folder] を選択します。
6. フォルダの名前 (例: **`favorite-pics`**) を入力します。次に [フォルダの作成] をクリックします。

フォルダの公開

パブリックフォルダまたはバケットが特別に必要な場合を除き、Amazon S3 フォルダまたはバケットへのすべてのパブリックアクセスをブロックすることをお勧めします。フォルダを公開すると、インターネット上の誰もがフォルダ内でグループ化されたすべてのオブジェクトを表示できます。

Amazon S3 コンソールでは、フォルダをパブリックにすることができます。データへのアクセスをプレフィックスで制限するバケットポリシーを作成して、フォルダを公開することもできます。詳細については、「Amazon S3 用 Identity and Access Management」を参照してください。

> **Warning**
>
> Amazon S3 コンソールでフォルダを公開した後は、非公開に戻すことはできません。代わりに、非公開にするパブリックフォルダ内の各オブジェクトのアクセス許可の設定で、オブジェクトへのパブリックアクセスを禁止する必要があります。詳細については、「ACL の設定」を参照してください。

トピック

- フォルダサイズの計算
- フォルダの削除

フォルダサイズの計算

このセクションでは、Amazon S3 コンソールを使用してフォルダのサイズを計算する方法について説明します。

フォルダのサイズを計算するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [Buckets] (バケット) リストで、フォルダが保存されているバケットの名前を選択します。
4. [Objects] (オブジェクト) リストで、フォルダの名前の横にあるチェックボックスをオンにします。
5. [Actions] (アクション) を選択し、[Calculate total size] (合計サイズを計算する) を選択します。

Note

ページから移動すると、フォルダ情報 (合計サイズを含む) は利用できなくなります。合計サイズを再度表示するには、計算をやり直す必要があります。

Important

- バケット内の指定されたオブジェクトまたはフォルダで [Calculate total size] (合計サイズを計算する) アクションを使用すると、Amazon S3 はオブジェクトの総数と合計ストレージサイズを計算します。ただし、不完全または進行中のマルチパートのアップロード、および以前のバージョンまたは最新ではないバージョンは、オブジェクトの総数または合計サイズには含まれません。このアクションは、バケットに保存されている各オブジェクトの現在のバージョンまたは最新バージョンのオブジェクトの総数と合計サイズのみを計算します。

  例えば、バケット内に 2 つのバージョンのオブジェクトがある場合、Amazon S3 のストレージ計算ツールはそれらを 1 つのオブジェクトとしてしかカウントしません。そのため、Amazon S3 コンソールで計算されるオブジェクトの総数は、S3 Storage Lens に表示される [Object Count] (オブジェクト数) メトリクスや Amazon CloudWatch メトリクス、`NumberOfObjects` によって報告される数と異なる場合があります。同様に、ストレージの合計サイズは、S3 Storage Lens に表示される [Total Storage] (ストレージの合計) メトリクスや CloudWatch に表示される `BucketSizeBytes` メトリクスと異なる場合があります。
- 大きなフォルダの合計サイズの計算に時間がかかりすぎる場合は、代わりに Amazon S3 Inventory と Amazon S3 Select を使用することを検討します。まず、S3 インベントリ設定を作成し、大きなフォルダの各オブジェクトのサイズメタデータをインベントリレポートに含めます。最初の S3 インベントリレポートが配信されるまでに最大 48 時間かかることがあります。インベントリレポートが公開されたら、S3 Select の `SUM` 式を使用してインベントリレポートをクエリし、フォルダ内のオブジェクトのサイズを集計します。詳細については、S3 コンソールを使用したインベントリの設定および SUM の例を参照してください。

フォルダの削除

このセクションでは、Amazon S3 コンソールを使用して S3 バケットからフォルダを削除する方法について説明します。

Amazon S3 の機能と料金の詳細については、Amazon S3 を参照してください。

S3 バケットからフォルダを削除するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、フォルダを削除するバケットの名前を選択します。
3. [オブジェクト] リストで、削除するフォルダとオブジェクトの横にあるチェックボックスをオンにします。
4. [削除] を選択します。
5. [オブジェクトの削除] ページで、削除ターゲットとして選択したフォルダの名前がリストされていることを確認します。
6. [オブジェクトの削除] ボックスに「**delete**」と入力し、[オブジェクトの削除] をクリックします。

> **Warning**
>
> このアクションは、指定されたすべてのオブジェクトを削除します。フォルダを削除する場合は、削除アクションが完了するのを待ってから、フォルダに新しいオブジェクトを追加します。そうしなければ、新しいオブジェクトも削除される可能性があります。

Amazon S3 コンソールでのオブジェクトの概要の表示

Amazon S3 コンソールを使用して、オブジェクトの概要を表示できます。コンソールでは、オブジェクトについてのすべての重要な情報をまとめて表示します。

オブジェクトの詳細ページを開くには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

2. [Buckets] (バケット) リストで、オブジェクトが含まれるバケットの名前を選択します。
3. [オブジェクト] リストで、概要を表示するオブジェクトの名前を選択します。

   オブジェクトの [詳細] ページが開きます。
4. オブジェクトをダウンロードするには、[Object actions] を選択し、[ダウンロード] をクリックします。クリップボードにオブジェクトのパスをコピーするには、[オブジェクト URL] で URL を選択します。
5. バケットでバージョニングが有効になっている場合、[Latest versions] を選択してオブジェクトのすべてのバージョンを表示します。

   - オブジェクトバージョンをダウンロードするには、バージョン ID の横にあるチェックボックスをオンにし、[アクション]、[ダウンロード] の順に選択します。
   - オブジェクトバージョンを削除するには、バージョン ID の横にあるチェックボックスをオンにし、[削除] をクリックします。

   > **Important**
   >
   > 最新 (現在) のバージョンとして削除された場合のみ、オブジェクトを復元できます。削除されたオブジェクトの以前のバージョンを復元することはできません。

Amazon S3 コンソールでのオブジェクトのプロパティの表示

Amazon S3 コンソールを使用して、ストレージクラス、暗号化設定、タグ、メタデータなどのオブジェクトのプロパティを表示できます。

オブジェクトのプロパティを表示するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets] (バケット) リストで、オブジェクトが含まれるバケットの名前を選択します。
3. [オブジェクト] リストで、プロパティを表示するオブジェクトの名前を選択します。

   オブジェクトの [オブジェクトの概要] が開きます。下にスクロールすると、オブジェクトのプロパティを表示できます。
4. [オブジェクトの概要] ページでは、以下のオブジェクトのプロパティを設定できます。

> **Note**
>
> - ストレージクラス、暗号化、メタデータなどのプロパティを変更すると、新しいオブジェクトが作成され、古いオブジェクトが置き換えられます。S3 バージョニングが有効になっている場合は、オブジェクトの新しいバージョンが作成され、既存のオブジェクトが古いバージョンになります。また、プロパティを変更するロールは、新しいオブジェクト (オブジェクトのバージョン) の所有者になります。
> - ユーザー定義タグを持つオブジェクトのストレージクラス、暗号化 、またはメタデータプロパティを変更する場合は、`s3:GetObjectTagging` アクセス許可が必要です。ユーザー定義タグがなく、サイズが 16 MB を超えるオブジェクトのプロパティを変更する場合は、`s3:GetObjectTagging` アクセス許可も必要です。
>
> ターゲットバケットポリシーによって `s3:GetObjectTagging` アクションが拒否されると、オブジェクトのこれらのプロパティは更新されますが、ユーザー定義タグはオブジェクトから削除され、エラーが発生します。

a. Storage class – Amazon S3 内にある各オブジェクトには、ストレージクラスが関連付けられています。オブジェクトにアクセスする頻度に応じて、使用するストレージクラスを選択します。S3 オブジェクトのデフォルトのストレージクラスは STANDARD です。オブジェクトをアップロードするときに使用するストレージクラス選択します。ストレージクラスの詳細については、Amazon S3 ストレージクラスを使用する を参照してください。

   オブジェクトのアップロード後にストレージクラスを変更するには、[ストレージクラス] を選択します。目的のストレージクラスを選択し、[保存] を選択します。

b. サーバー側の暗号化設定 - サーバー側の暗号化を使用して S3 オブジェクトを暗号化できます。詳細については、AWS KMS (SSE-KMS) によるサーバー側の暗号化の指定 または Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) の指定 を参照してください。

c. Metadata – Amazon S3 内の各オブジェクトには、それぞれのメタデータを表す一連の名前/値ペアがあります。S3 オブジェクトにメタデータを追加する方法については、「Amazon S3 コンソールでのオブジェクトメタデータの編集」を参照してください。

d. タグ - S3 オブジェクトにタグを追加して、ストレージを分類します。詳細については、「タグを使用してストレージを分類する」を参照してください。

e. Object Lock のリーガルホールドと保持 - オブジェクトが削除されないようにできます。詳細については、「S3 オブジェクトロックの使用」を参照してください。

署名付き URL の使用

バケットポリシーを更新せずに、Amazon S3 内のオブジェクトへの時間制限付きのアクセス権を付与するには、署名付き URL を使用できます。署名付き URL をブラウザに入力するか、プログラムで使用してオブジェクトをダウンロードできます。署名付き URL で使用される認証情報は、URL を生成した AWS ユーザーのものです。

また、署名付き URL を使用して、Amazon S3 バケットに対する特定のオブジェクトのアップロードを他のユーザーに許可することもできます。これにより、他のユーザーは AWS のセキュリティ認証情報やアクセス許可を持たなくてもアップロードできます。署名付き URL で指定したのと同じキーを持つオブジェクトがバケット内に既に存在する場合、Amazon S3 は既存のオブジェクトをアップロードしたオブジェクトで置き換えます。

署名付き URL は、有効期限日時まで複数回使用できます。

署名付き URL を作成する場合には、ご自身のセキュリティ認証情報を設定し、さらに次の情報を指定する必要があります。

- Amazon S3 バケット
- オブジェクトキー (オブジェクトのダウンロード先は Amazon S3 バケット、アップロード先はアップロード先のファイル名)
- HTTP メソッド (オブジェクトをダウンロードする場合は `GET`、アップロードする場合は `PUT`)
- 有効期限の時間間隔

現在、Amazon S3 の署名付き URL では、オブジェクトをアップロードする際に次のデータ整合性チェックサムアルゴリズム (CRC32、CRC32C、SHA-1、SHA-256) を使用することはできません。アップロード後にオブジェクトの整合性を確認するには、署名付き URL でアップロードするときに、オブジェクトの MD5 ダイジェストを指定できます。オブジェクトの整合性の詳細については、「オブジェクトの整合性をチェックする」を参照してください。

トピック

- 署名付き URL を作成できるユーザー
- 署名付き URL の有効期限

- 署名付き URL 機能の制限
- 署名付き URL を使用したオブジェクトの共有
- 署名付き URL を使用したオブジェクトのアップロード

署名付き URL を作成できるユーザー

有効なセキュリティ認証情報を持つすべてのユーザーが、署名付き URL を作成できます。しかし、何らかの理由でオブジェクトに正常にアクセスするには、署名付き URL を使用して行うオペレーションの実行許可を持っているユーザーが、署名付き URL を作成する必要があります。

署名付き URL の作成に使用できる認証情報のタイプは以下のとおりです。

- IAM インスタンスプロファイル – 最大 6 時間有効。
- AWS Security Token Service — 長期のセキュリティ認証情報を使用して署名した場合、または一時的な認証情報の有効期間 (どちらか早い方) で最長 36 時間有効です。
- IAM ユーザー – AWS Signature Version 4 を使用している場合は、最大 7 日間まで有効。

  最大 7 日間有効な署名付き URL を作成するには、まず、署名付き URL の作成に使用するメソッドへの IAM ユーザー認証情報 (アクセスキーとシークレットキー) の委任を行います。

> **Note**
>
> 一時的な認証情報を使用して署名付き URL を作成した場合、その認証情報が有効期限切れになると、URL は失効します。一般に、署名付き URL は、作成に使用した認証情報が取り消された場合、削除された場合、または非アクティブ化された場合は、期限切れになります。URL の有効期限がより長い場合でも失効します。一時的なセキュリティ認証情報の有効期間については、IAM ユーザーガイドの「AWS STS API オペレーションの比較」を参照してください。

署名付き URL の有効期限

署名付き URL は、URL の生成時に指定した期間にわたって有効です。Amazon S3 コンソールで署名付き URL を作成した場合、有効期限は 1 分から 12 時間の間で設定できます。AWS CLI または AWS SDK を使用する場合、有効期限は最大 7 日間に設定できます。

一時トークンを使用して署名付き URL を作成した場合、そのトークンが有効期限切れになると、URL は期限切れになります。一般に、署名付き URL は、作成に使用した認証情報が取り消された場合、削除された場合、または非アクティブ化された場合は、期限切れになります。URL の有効期限がより長い場合でも失効します。認証情報の使用が有効期限にどのように影響するかについては、「署名付き URL を作成できるユーザー」を参照してください。

Amazon S3 は、HTTP リクエスト時に署名付き URL の有効期限日時を確認します。例えば、有効期限が切れる時刻の直前にクライアントが大きなファイルのダウンロードを開始した場合は、ダウンロード中に有効期限時刻が経過しても、そのダウンロードは継続されます。しかし、接続が中断し、クライアントがダウンロードを再開しようとした時点で有効期限切れの時刻が経過している場合には、そのダウンロードは失敗します。

署名付き URL 機能の制限

署名付き URL の機能は、それを作成したユーザーの許可によって制限されます。本質的に署名付き URL は、それらを保有しているユーザーに対しアクセスを許可するためのベアラートークンです。そのため、適切に保護することをお勧めします。署名付き URL の使用を制限するために使用できるいくつかの方法を以下に示します。

AWS Signature Version 4 (SigV4)

署名済み URL リクエストが AWS Signature Version 4 (SigV4) により認証される際に実行する特定の動作を適用するには、バケットポリシーとアクセスポイントポリシーで条件キーを使用します。例えば、次のバケットポリシーでは、`s3:signatureAge` 条件を使用することで、署名が作成後 10 分を超えている場合、*`example-s3-bucket1`* バケット内のオブジェクトに対する Amazon S3 の署名付き URL リクエストをすべて拒否します。この例を実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Deny a presigned URL request if the signature is more than 10 min
 old",
         "Effect": "Deny",
         "Principal": {"AWS":"*"},
         "Action": "s3:*",
         "Resource": "arn:aws:s3:::example-s3-bucket1/*",
         "Condition": {
            "NumericGreaterThan": {
```

```
                "s3:signatureAge": 600000
            }
        }
    }
  ]
}
```

AWS Signature Version 4 に関連するポリシーキーの詳細については、「Amazon Simple Storage Service API リファレンス」の「AWS Signature Version 4 認証固有のポリシーキー」を参照してください。

ネットワークパスでの制限

署名付き URL の使用と、特定のネットワークパスへのすべての Amazon S3 アクセスを制限する場合は、AWS Identity and Access Management (IAM) ポリシーを記述できます。これらのポリシーは、呼び出しを作成する IAM プリンシパル、Simple Storage Service (Amazon S3) バケット、またはその両方に設定できます。

IAM プリンシパルでのネットワークパスの制限では、これらの認証情報のユーザーは、指定したネットワークからリクエストを送信する必要があります。バケットまたはアクセスポイントの制限により、そのリソースに対するすべてのリクエストは、指定したネットワークから発信される必要があります。これらの制限は、署名付き URL のシナリオ以外でも適用されます。

どの IAM グローバル条件キーを使用するかは、エンドポイントのタイプによって異なります。Amazon S3 のパブリックエンドポイントを使用している場合は、`aws:SourceIp` を使用します。Amazon S3 への仮想プライベートクラウド (VPC) エンドポイントを使用している場合は、`aws:SourceVpc` または `aws:SourceVpce` を使用します。

次の IAM ポリシーステートメントでは、プリンシパルは、指定されたネットワーク範囲からのみ AWS にアクセスする必要があります。このポリシーステートメントでは、すべてのアクセスがその範囲から発信される必要があります。これは、Amazon S3 の署名付き URL を使用しているユーザーにも当てはまります。この例を実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
{
    "Sid": "NetworkRestrictionForIAMPrincipal",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
```

```
        "NotIpAddressIfExists": {"aws:SourceIp": "IP-address-range"},
        "BoolIfExists": {"aws:ViaAWSService": "false"}
    }
}
```

aws:SourceIp AWS グローバル条件キーを使用して Amazon S3 バケットへのアクセスを特定のネットワーク範囲に制限する他のバケットポリシーの例については、「特定の IP アドレスに基づくアクセス管理」を参照してください。

署名付き URL を使用したオブジェクトの共有

デフォルトでは、すべての Amazon S3 オブジェクトはプライベートであり、オブジェクトの所有者のみがアクセスできます。ただし、オブジェクトの所有者は、署名付き URL を作成することで、他のユーザーとオブジェクトを共有できます。署名済み URL は、セキュリティ認証情報を使用してオブジェクトをダウンロードするアクセス許可を期限付きで付与します。署名付き URL をブラウザに入力するか、プログラムで使用してオブジェクトをダウンロードできます。署名付き URL で使用される認証情報は、URL を生成した AWS ユーザーのものです。

署名付き URL の一般的な情報については、「署名付き URL の使用」を参照してください。

オブジェクトを共有するための署名付き URL は、Amazon S3 コンソール、AWS Explorer for Visual Studio (Windows)、または AWS Toolkit for Visual Studio Code を使用してコードを記述せずに作成できます。AWS Command Line Interface (AWS CLI) または AWS SDK を使用して、署名付き URL をプログラムで生成することもできます。

S3 コンソールの使用

Amazon S3 コンソールで次の手順に従い、オブジェクトを共有するための署名付き URL を生成できます。コンソールを使用した場合、署名付き URL の最大有効期限は作成時点から 12 時間です。

Amazon S3 コンソールを使用して署名付き URL を生成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、署名付き URL を取得するオブジェクトが含まれているバケットの名前を選択します。
4. [オブジェクト] リストで、署名付き URL を作成するオブジェクトを選択します。

5. [オブジェクトアクション] メニューで、[署名付き URL で共有] を選択します。
6. 署名付き URL の有効期間を指定します。
7. [Create presigned URL] (署名付き URL を作成) を選択します。
8. 確認が表示されると、URL は自動的にクリップボードにコピーされます。署名済み URL を再度コピーする必要がある場合は、署名済み URL をコピーするボタンが表示されます。

AWS CLI の使用

次の例では、AWS CLI コマンドにより、Amazon S3 バケットのオブジェクトを共有するための署名付き URL を生成します。AWS CLI を使用する場合、署名付き URL の最大有効期限は、作成時点から 7 日間です。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3 presign s3://example-s3-bucket1/mydoc.txt --expires-in 604800
```

> **Note**
>
> 2019 年 3 月 20 日より後に開設されたすべての AWS リージョンでは、リクエストで endpoint-url と AWS #####を指定する必要があります。すべての Amazon S3 のリージョンとエンドポイントのリストについては、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。

```
aws s3 presign s3://example-s3-bucket1/mydoc.txt --expires-in 604800 --region af-south-1 --endpoint-url https://s3.af-south-1.amazonaws.com
```

詳細については、AWS CLI コマンドリファレンスの「presign」を参照してください。

AWS SDK の使用

AWS SDK を使用してオブジェクトを共有するための署名付き URL を生成する例については、「AWS SDK を使用して Amazon S3 の署名付き URL を作成する」を参照してください。

署名付き URL の生成に AWS SDK を使用した場合、最大有効期限は、作成時点から 7 日間となります。

> **Note**
>
> 2019 年 3 月 20 日より後に開設されたすべての AWS リージョンでは、リクエストで `endpoint-url` と `AWS #####`を指定する必要があります。すべての Amazon S3 のリージョンとエンドポイントのリストについては、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。

> **Note**
>
> AWS SDK を使用する場合、Tagging 属性はクエリパラメータではなく、ヘッダーでなければなりません。他のすべての属性は、署名済み URL のパラメータとして渡すことができます。

AWS Toolkit for Visual Studio の使用 (Windows)

> **Note**
>
> 現時点では、AWS Toolkit for Visual Studio は Visual Studio for Mac をサポートしていません。

1. 「AWS Toolkit for Visual Studio ユーザーガイド」の「Toolkit for Visual Studio のインストールとセットアップ」の手順に従って、AWS Toolkit for Visual Studio をインストールします。
2. 「AWS Toolkit for Visual Studio ユーザーガイド」の「AWS に接続する」の手順に従って、AWS に接続します。
3. AWS Explorer というラベルの左側のパネルで、オブジェクトを含むバケットをダブルクリックします。
4. 署名付き URL を生成するオブジェクトを右クリックし、[署名付き URL を作成...] を選択します。
5. ポップアップウィンドウで、署名付き URL の有効期限日時を設定します。
6. [オブジェクトキー] は、選択したオブジェクトに基づいて事前に入力されている必要があります。
7. [GET] を選択して、この署名付き URL をオブジェクトのダウンロードに使用するように指定します。

8. [生成] ボタンを選択します。

9. URL をクリップボードにコピーするには、[コピー] を選択します。

10生成された署名付き URL を使用するには、URL を任意のブラウザに貼り付けます。

AWS Toolkit for Visual Studio Code の使用

Visual Studio Code を使用している場合は、AWS Toolkit for Visual Studio Code を使用してコードを一切記述せずに、オブジェクトを共有するための署名付き URL を生成できます。一般的な情報については、「AWS Toolkit for Visual Studio Code ユーザーガイド」の「AWS Toolkit for Visual Studio Code」を参照してください。

AWS Toolkit for Visual Studio Code のインストール手順については、「AWS Toolkit for Visual Studio Code ユーザーガイド」の「AWS Toolkit for Visual Studio Code のインストール」を参照してください。

1. 「AWS Toolkit for Visual Studio Code ユーザーガイド」の「AWS Toolkit for Visual Studio Code に接続する」の手順に従って、AWS に接続します。
2. Visual Studio Code の左側のパネルで AWS ロゴを選択します。
3. [エクスプローラー] で、[S3] を選択します。
4. バケットとファイルを選択し、右クリックしてコンテキストメニューを開きます。
5. [署名済み URL を生成] を選択し、有効期限 (分単位) を設定します。
6. Enter キーを押すと、署名付き URL がクリップボードにコピーされます。

署名付き URL を使用したオブジェクトのアップロード

署名付き URL を使用して、自分の Amazon S3 バケットにオブジェクトをアップロードすることを他のユーザーに許可できます。署名付き URL を使用すると、他のユーザーは AWS セキュリティ認証情報やアクセス許可を持っていなくてもアップロードできます。署名付き URL は、それを作成したユーザーのアクセス許可によって制限されます。つまり、オブジェクトをアップロードするための署名付き URL を受け取った場合、オブジェクトをアップロードできるのは、署名付き URL の作成者がそのオブジェクトのアップロードに必要なアクセス許可を持っている場合のみです。

この URL を使用してオブジェクトをアップロードすると、Amazon S3 は指定されたバケットにオブジェクトを作成します。署名付き URL で指定したのと同じキーを持つオブジェクトがバケット内

に既存する場合、Amazon S3 は既存のオブジェクトをアップロードしたオブジェクトで置き換えます。アップロード後は、バケット所有者がオブジェクトを所有します。

署名付き URL の一般的な情報については、「署名付き URL の使用」を参照してください。

AWS Explorer for Visual Studio を使用すると、コードを一切記述せずに、オブジェクトをアップロードするための署名付き URL を作成できます。AWS SDK を使用して署名付き URL をプログラムで生成することもできます。

AWS Toolkit for Visual Studio の使用 (Windows)

> **Note**
>
> 現時点では、AWS Toolkit for Visual Studio は Visual Studio for Mac をサポートしていません。

1. 「AWS Toolkit for Visual Studio ユーザーガイド」の「Toolkit for Visual Studio のインストールとセットアップ」の手順に従って、AWS Toolkit for Visual Studio をインストールします。
2. 「AWS Toolkit for Visual Studio ユーザーガイド」の「AWS に接続する」の手順に従って、AWS に接続します。
3. [AWS Explorer] というラベルの左側のパネルで、オブジェクトをアップロードするバケットを右クリックします。
4. [署名付き URL を作成] を選択します。
5. ポップアップウィンドウで、署名付き URL の有効期限日時を設定します。
6. [オブジェクトキー] で、アップロードするファイルの名前を設定します。アップロードするファイルは、この名前と正確に一致する必要があります。バケットに同じキーを持つオブジェクトがバケット内に既に存在する場合、Amazon S3 は既存のオブジェクトを新しくアップロードしたオブジェクトで置き換えます。
7. [PUT] を選択し、この署名付き URL をオブジェクトのアップロードに使用するように指定します。
8. [生成] ボタンを選択します。
9. URL をクリップボードにコピーするには、[コピー] を選択します。
10. この URL を使用するには、`curl` コマンドを使って PUT リクエストを送信できます。ファイルへのフルパスと署名付き URL 自体を含めます。

```
curl -X PUT -T "/path/to/file" "presigned URL"
```

AWS SDK の使用

AWS SDK を使用してオブジェクトをアップロードするための署名付き URL を生成する例については、「AWS SDK を使用して Amazon S3 の署名付き URL を作成する」を参照してください。

署名付き URL の生成に AWS SDK を使用した場合、最大有効期限は、作成時点から 7 日間となります。

> **Note**
>
> 2019 年 3 月 20 日より後に開設されたすべての AWS リージョンでは、リクエストで `endpoint-url` と `AWS #####`を指定する必要があります。すべての Amazon S3 のリージョンとエンドポイントのリストについては、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。

S3 Object Lambda を使用したオブジェクトの変換

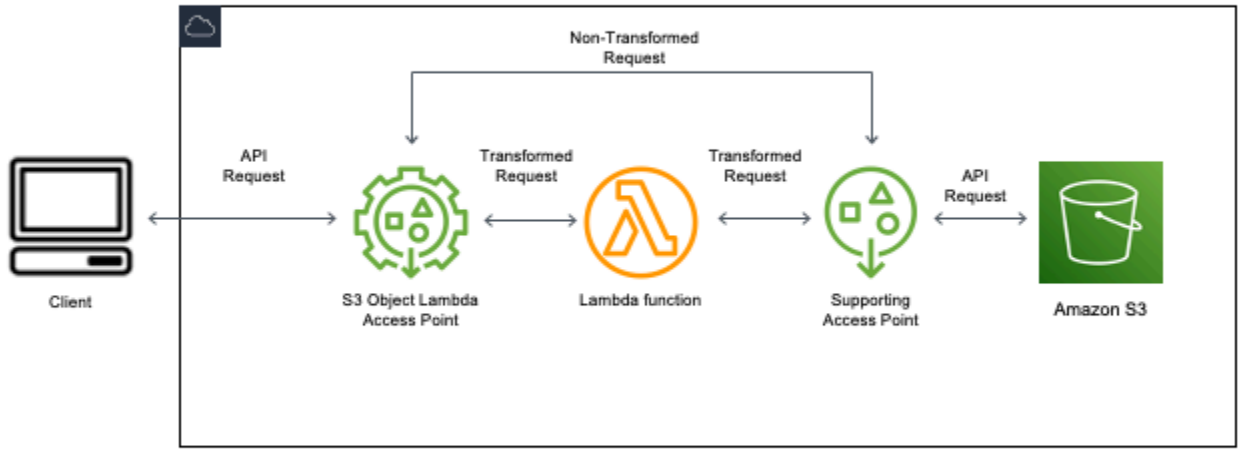
Amazon S3 Object Lambda を使用すると、Amazon S3 `GET`、`LIST`、`HEAD` リクエストに独自のコードを追加して、データがアプリケーションに返されるときにそのデータを変更および処理できます。カスタムコードを使用すると、S3 `GET` リクエストによって返されるデータを変更し、行のフィルタリング、画像の動的なサイズ変更と透かし、機密データの編集などを行うことができます。また、S3 Object Lambda を使用して、バケット内のすべてのオブジェクトのカスタムビューを作成する S3 `LIST` リクエストの出力や、オブジェクト名やサイズなどのオブジェクトメタデータを変更する S3 `HEAD` リクエストの出力を変更することもできます。S3 Object Lambda を Amazon CloudFront ディストリビューションのオリジンとして使用して、画像の自動サイズ変更、古い形式 (JPEG から WebP など) のトランスコーディング、メタデータの削除など、エンドユーザー向けにデータを調整できます。詳細については、AWS ブログの「Use Amazon S3 Object Lambda with Amazon CloudFront」(Amazon S3 Object Lambda を Amazon CloudFront で使用する) を参照してください。AWS Lambda 関数を利用しているため、コードは AWS によって完全に管理されたインフラストラクチャ上で実行されます。S3 Object Lambda を使用すると、データの派生コピーを作成して保存したり、プロキシを実行したりする必要が少なくなり、アプリケーションに変更を加える必要もなくなります。

S3 オブジェクト Lambda の仕組み

S3 Object Lambda は AWS Lambda 関数を使用して、標準の S3 GET、LIST、HEAD リクエストの出力を自動的に処理します。AWS Lambda は、基盤となるコンピューティングリソースの管理を必要とせずに、顧客定義のコードを実行するサーバーレスコンピューティングサービスです。データ変換を特定のユースケースに合わせて調整し、独自のカスタム Lambda 関数を作成して実行することができます。

Lambda 関数を設定したら、Object Lambda アクセスポイントと呼ばれる S3 Object Lambda サービスエンドポイントにアタッチします。Object Lambda アクセスポイントは、サポートアクセスポイントと呼ばれる、標準の S3 アクセスポイントを使用して Amazon S3 にアクセスします。

Object Lambda アクセスポイントにリクエストを送信すると、Amazon S3 は自動的に Lambda 関数を呼び出します。その後、S3 Object Lambda アクセスポイントを介して S3 GET、LIST、または HEAD リクエストを使用して取得されたデータは、変換された結果をアプリケーションに返します。次の図に示すように、他のすべてのリクエストは通常どおりに処理されます。



このセクションのトピックでは、S3 Object Lambda の使用方法について説明します。

トピック

Object Lambda アクセスポイントの作成

Object Lambda アクセスポイントは、厳密に 1 つの標準アクセスポイントと 1 つの Amazon S3 バケットに関連付けられます。Object Lambda アクセスポイントを作成するには、次のリソースが必要です。

- Amazon S3 バケット。バケットの作成については、「the section called “バケットの作成”」を参照してください。
- 標準 S3 アクセスポイント。Object Lambda アクセスポイントを使用する場合、この標準アクセスポイントはサポートアクセスポイントと呼ばれます。標準アクセスポイントの作成については、「the section called “アクセスポイントの作成”」を参照してください。
- AWS Lambda 関数。独自の Lambda 関数を作成することも、事前に構築された関数を使用することもできます。Lambda 関数の詳細については、「the section called “Lambda 関数の記述”」を参照してください。事前構築された関数の詳細については、「AWS で構築された Lambda 関数の使用」を参照してください。
- (オプション) AWS Identity and Access Management (IAM) ポリシー。Amazon S3 アクセスポイントは IAM リソースポリシーをサポートしています。これにより、リソース、ユーザー、またはその他の条件別にアクセスポイントの使用を制御できます。これらのポリシーの作成の詳細については、「the section called “IAM ポリシーの設定”」を参照してください。

以下のセクションでは、下記を使用して Object Lambda アクセスポイントを作成する方法について説明します。

- AWS Management Console
- AWS Command Line Interface (AWS CLI)

- AWS CloudFormation テンプレート
- AWS Cloud Development Kit (AWS CDK)

REST API を使用して Object Lambda アクセスポイントを作成する方法については、Amazon Simple Storage Service API リファレンスの「CreateAccessPointForObjectLambda」を参照してください。

Object Lambda アクセスポイントの作成

次のいずれかの手順に従って Object Lambda アクセスポイントを作成します。

S3 コンソールの使用

コンソールを使用して Object Lambda のアクセスポイントを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションバーで、現在表示されている AWS リージョン の名前をクリックします。次に、切り替え先のリージョンを選択します。
3. 左のナビゲーションペインで、[Object Lambda アクセスポイント] を選択します。
4. [Object Lambda アクセスポイント] ページで、[Object Lambda アクセスポイントの作成] を選択します。
5. [Object Lambda Access Point name] (Object Lambda アクセスポイント名) に、アクセスポイントに使用する名前を入力します。

   標準のアクセスポイントと同様に、Object Lambda には命名規則があります。詳細については、「Amazon S3 アクセスポイントの命名規則」を参照してください。
6. [Supporting Access Point] (サポートアクセスポイント) で、使用する標準アクセスポイントを入力または参照します。アクセスポイントは、変換するオブジェクトと同じ AWS リージョン に存在する必要があります。標準アクセスポイントの作成については、「the section called “アクセスポイントの作成”」を参照してください。
7. [変換の設定] で、Object Lambda アクセスポイントのデータを変換する関数を追加できます。次のいずれかを行います。

   - アカウントに既に AWS Lambda 関数がある場合は、[Invoke Lambda function] (Lambda 関数の呼び出し) で選択できます。ここで、AWS アカウント で Lambda 関数の Amazon リソースネーム (ARN) を入力するか、ドロップダウンメニューから Lambda 関数を選択できます。

- AWS で構築された関数を使用する場合は、[AWS で構築された関数] で関数名を選択し、[Lambda 関数を作成] を選択します。これにより Lambda コンソールが表示され、そこで構築した関数を AWS アカウント にデプロイできます。構築した関数の詳細については、「AWS で構築された Lambda 関数の使用」を参照してください。

[S3 APIs] で、呼び出す API オペレーションを 1 つ以上選択します。選択した各 API に対して、呼び出す Lambda 関数を指定する必要があります。

8. (オプション) [Payload] (ペイロード) に、入力として Lambda 関数に提供する JSON テキストを追加します。同じ Lambda 関数を呼び出す異なる Object Lambda アクセスポイントに対して、異なるパラメータを使用してペイロードを設定できます。これにより、Lambda 関数の柔軟性が向上します。

> **Important**
>
> Object Lambda アクセスポイントを使用する場合、ペイロードには機密情報を一切含めないでください。

9. (オプション) [Range and part number] (範囲とパート番号) では、範囲とパート番号ヘッダーで GET および HEAD リクエストを処理するには、このオプションを有効にする必要があります。このオプションを有効にすると、Lambda 関数がこれらのリクエストを認識して処理できることが確認されます。範囲ヘッダーとパート番号の詳細については、Range および partNumber ヘッダーの操作 を参照してください。
10. (オプション) [リクエストのメトリクス] で、[有効化] または [無効化] を選択して、Amazon S3 モニタリングを Object Lambda アクセスポイントに追加します。リクエストメトリクスには、Amazon CloudWatch の標準料金が課金されます。
11. (オプション) [Object Lambda Access Point policy] (Object Lambda アクセスポイントポリシー) で、リソースポリシーを設定します。リソースポリシーは、指定された Object Lambda アクセスポイントにアクセス許可を付与し、リソース、ユーザー、その他の条件別にアクセスポイントの使用を制御します。Object Lambda アクセスポイントのリソースポリシーの詳細については、「Object Lambda アクセスポイントの IAM ポリシーの設定」を参照してください。
12. [Block Public Access settings for this Object Lambda Access Point] (この Object Lambda アクセスポイントのパブリックアクセス設定をブロック) で、適用するパブリックアクセスブロック設定を選択します。新しい Object Lambda アクセスポイントでは、すべてのパブリックアクセスブロック設定がデフォルトで有効になります。デフォルトの設定のままにしておくことをお勧めします。Amazon S3 は、現在、Object Lambda アクセスポイントの作成後における Object Lambda アクセスポイントのパブリックアクセスブロック設定の変更をサポートしていません。

Amazon S3 パブリックアクセスブロックの詳細については、「アクセスポイントへのパブリックアクセスの管理」を参照してください。

13. [Create Object Lambda Access Point] (Object Lambda アクセスポイントの作成) を選択します。

AWS CLI の使用

AWS CloudFormation テンプレートを使用して Object Lambda アクセスポイントを作成するには

> **Note**
>
> 次のコマンドを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

1. S3 Object Lambda のデフォルト設定で、AWS Lambda 関数デプロイパッケージ `s3objectlambda_deployment_package.zip` をダウンロードします。
2. 次の `put-object` コマンドを実行して、Amazon S3 バケットにパッケージをアップロードします。

```
aws s3api put-object --bucket Amazon S3 bucket name --key
 s3objectlambda_deployment_package.zip --body release/
s3objectlambda_deployment_package.zip
```

3. S3 Object Lambda のデフォルト設定で AWS CloudFormation テンプレート `s3objectlambda_defaultconfig.yaml` をダウンロードします。
4. 次の `deploy` コマンドを実行して、テンプレートを AWS アカウント にデプロイします。

```
aws cloudformation deploy --template-file s3objectlambda_defaultconfig.yaml \
 --stack-name AWS CloudFormation stack name \
 --parameter-overrides ObjectLambdaAccessPointName=Object Lambda Access Point name
 \
  SupportingAccessPointName=Amazon S3 access point S3BucketName=Amazon S3 bucket \
  LambdaFunctionS3BucketName=Amazon S3 bucket containing your Lambda package \
  LambdaFunctionS3Key=Lambda object key LambdaFunctionS3ObjectVersion=Lambda object
 version \
  LambdaFunctionRuntime=Lambda function runtime --capabilities capability_IAM
```

GET、HEAD、および LIST API オペレーションで Lambda を呼び出すようにこの AWS CloudFormation テンプレートを設定できます。テンプレートのデフォルト設定の変更の詳細については、「the section called “AWS CloudFormation を使用して、S3 Object Lambda のセットアップを自動化します。”」を参照してください。

AWS CLI を使用して Object Lambda アクセスポイントを作成するには

> **Note**
>
> 次のコマンドを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

以下の例では、アカウント *111122223333* のバケット *DOC-EXAMPLE-BUCKET1* に「*my-object-lambda-ap*」という名前の Object Lambda アクセスポイントを作成します。この例では、「*example-ap*」という名前の標準アクセスポイントが既に作成されていることを前提としています。標準アクセスポイントの作成については、the section called “アクセスポイントの作成” を参照してください。

この例では、AWS の事前構築された関数 decompress を使用しています。事前構築された関数の詳細については、「the section called “AWS で構築された関数の使用”」を参照してください。

1. バケットを作成します。この例では、*DOC-EXAMPLE-BUCKET1* を使用します。バケットの作成については、「the section called “バケットの作成”」を参照してください。
2. 標準アクセスポイントを作成し、バケットにアタッチします。この例では、*example-ap* を使用します。標準アクセスポイントの作成については、「the section called “アクセスポイントの作成”」を参照してください。
3. 次のいずれかを行います。

   - Amazon S3 オブジェクトの変換に使用する Lambda 関数をアカウント内に作成します。Lambda 関数の詳細については、「the section called “Lambda 関数の記述”」を参照してください。AWS CLI でカスタム関数を使用するには、「AWS Lambda 開発者ガイド」の「AWS CLI での Lambda の使用」を参照してください。
   - AWS の事前構築された Lambda 関数を使用します。事前構築された関数の詳細については、「AWS で構築された Lambda 関数の使用」を参照してください。
4. my-olap-configuration.json という名前の JSON 設定ファイルを作成します。この設定では、前のステップで作成した Lambda 関数のサポートアクセスポイントと Amazon リソースネーム (ARN)、または使用している事前構築された関数の ARN を指定します。

Example

```
{
    "SupportingAccessPoint" : "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
    "TransformationConfigurations": [{
        "Actions" : ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
        "ContentTransformation" : {
            "AwsLambda": {
                "FunctionPayload" : "{\"compressionType\":\"gzip\"}",
                "FunctionArn" : "arn:aws:lambda:us-east-1:111122223333:function/
compress"
            }
        }
    }]
}
```

5. `create-access-point-for-object-lambda` コマンドを実行して Object Lambda アクセスポイントを作成します。

```
aws s3control create-access-point-for-object-lambda --account-id 111122223333 --
name my-object-lambda-ap --configuration file://my-olap-configuration.json
```

6. (オプション) `my-olap-policy.json` という名前の JSON ポリシーファイルを作成します。

   Object Lambda アクセスポイントのリソースポリシーを追加することで、リソース、ユーザー、その他の条件別にアクセスポイントの使用を制御することができます。このリソースポリシーは、アカウント *444455556666* に指定された Object Lambda アクセスポイントへの `GetObject` 許可を付与します。

   Example

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "Grant account 444455556666 GetObject access",
            "Effect": "Allow",
            "Action": "s3-object-lambda:GetObject",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::444455556666:root"
            },
            "Resource": "your-object-lambda-access-point-arn"
        }
    ]
}
```

7. (オプション) put-access-point-policy-for-object-lambda コマンドを実行して、リソースポリシーを設定します。

```
aws s3control put-access-point-policy-for-object-lambda --account-id 111122223333
 --name my-object-lambda-ap --policy file://my-olap-policy.json
```

8. (オプション) ペイロードを規定します。

   ペイロードは、AWS Lambda 関数に入力として提供できるオプションの JSON です。同じ Lambda 関数を呼び出す異なる Object Lambda アクセスポイントに対して、異なるパラメータを使用してペイロードを設定できます。これにより、Lambda 関数の柔軟性が向上します。

   次の Object Lambda アクセスポイントの設定は、2 つのパラメータを持つペイロードを示しています。

```
{
 "SupportingAccessPoint": "AccessPointArn",
 "CloudWatchMetricsEnabled": false,
 "TransformationConfigurations": [{
  "Actions": ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
  "ContentTransformation": {
   "AwsLambda": {
    "FunctionArn": "FunctionArn",
    "FunctionPayload": "{\"res-x\": \"100\",\"res-y\": \"100\"}"
   }
  }
 }]
}
```

   次の Object Lambda アクセスポイント設定は、パラメータが 1 つあり、GetObject-Range、GetObject-PartNumber、HeadObject-Range、および HeadObject-PartNumber が有効になっているペイロードを示しています。

```
{
```

```
    "SupportingAccessPoint":"AccessPointArn",
    "CloudWatchMetricsEnabled": false,
    "AllowedFeatures": ["GetObject-Range", "GetObject-PartNumber", "HeadObject-
Range", "HeadObject-PartNumber"],
    "TransformationConfigurations": [{
        "Action": ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
        "ContentTransformation": {
            "AwsLambda": {
                "FunctionArn":"FunctionArn",
                "FunctionPayload": "{\"compression-amount\": \"5\"}"
            }
        }
    }]
}
```

> **Important**
>
> Object Lambda アクセスポイントを使用する場合、ペイロードには機密情報を一切含めないでください。

AWS CloudFormation コンソールとテンプレートの使用。

Amazon S3 が提供するデフォルト設定を使用して、Object Lambda アクセスポイントを作成できます。GitHub リポジトリ から AWS CloudFormation テンプレートと Lambda 関数のソースコードのリソースをダウンロードして、これらのリソースをデプロイして、Object Lambda アクセスポイントを設定します。

AWS CloudFormation テンプレートのデフォルト設定の変更の詳細については、「the section called “AWS CloudFormation を使用して、S3 Object Lambda のセットアップを自動化します。”」を参照してください。

テンプレートなしで AWS CloudFormation を使用する Object Lambda アクセスポイントの設定の詳細については、AWS CloudFormation ユーザーガイドの「AWS::S3ObjectLambda::AccessPoint」を参照してください。

Lambda 関数のデプロイパッケージをアップロードします

1. S3 Object Lambda のデフォルト設定で、AWS Lambda 関数デプロイパッケージ `s3objectlambda_deployment_package.zip` をダウンロードします。
2. Amazon S3 バケットにパッケージをアップロードします。

AWS CloudFormation コンソールを使用して Object Lambda のアクセスポイントを作成するには

1. S3 Object Lambda のデフォルト設定で AWS CloudFormation テンプレート `s3objectlambda_defaultconfig.yaml` をダウンロードします。
2. AWS マネジメントコンソール にサインインした後、https://console.aws.amazon.com/cloudformation/ で AWS CloudFormation コンソールを開きます。
3. 次のいずれかを行います。
   - AWS CloudFormation を初めて使用する場合は、AWS CloudFormation ホームページで [Create stack] (スタックの作成) を選択します。
   - 以前に AWS CloudFormation を使用したことがある場合は、左側のナビゲーションペインで [Stacks] (スタック) を選択します。[Create stack] (スタックの作成) を選択し、[With new resources (standard)] (新しいリソースを使用 (標準)) を選択します。
4. 前提条件 – テンプレートの準備で、テンプレートの準備完了を選択します。
5. [Specify template] (テンプレートの指定) で、[Upload a template file] (テンプレートファイルのアップロード) を選択して `s3objectlambda_defaultconfig.yaml` をアップロードします。
6. [Next] を選択します。
7. スタックの詳細の指定ページで、スタックの名前を入力します。
8. [Parameters] (パラメータ) セクションで、スタックテンプレートで定義されている次のパラメータを指定します。
   a. [CreateNewSupportingAccessPoint] では、次のいずれかを実行します。
      - テンプレートをアップロードした S3 バケット用のサポートアクセスポイントが既にある場合は、[false] を選択します。
      - このバケット用に新しいアクセスポイントを作成する場合は、[true] を選択します。
   b. [EnableCloudWatchMonitoring] では、Amazon CloudWatch リクエストメトリクスとアラームを有効にするかどうかに応じて、[true] または [false] を選択します。
   c. (オプション) [LambdaFunctionPayload] に、入力として Lambda 関数に提供する JSON テキストを追加します。同じ Lambda 関数を呼び出す異なる Object Lambda アクセスポイントに対して、異なるパラメータを使用してペイロードを設定できます。これにより、Lambda 関数の柔軟性が向上します。

> **Important**
>
> Object Lambda アクセスポイントを使用する場合、ペイロードには機密情報を一切含めないでください。

d. [LambdaFunctionRuntime] で、Lambda 関数に希望するランタイムを入力します。使用可能な選択は次のとおりです。`nodejs14.x`、`python3.9`、`java11`。

e. [LambdaFunctionS3BucketName] で、デプロイパッケージをアップロードした Amazon S3 バケット名を入力します。

f. [LambdaFunctionS3Key] で、デプロイパッケージをアップロードした Amazon S3 オブジェクトキーを入力します。

g. [LambdaFunctionS3ObjectVersion] で、デプロイパッケージをアップロードした Amazon S3 オブジェクトバージョンを入力します。

h. [ObjectLambdaAccessPointName] で、Object Lambda アクセスポイント名を入力します。

i. [S3BucketName] で、Object Lambda アクセスポイントと関連付けられる Amazon S3 のバケット名を入力します。

j. [SupportingAccessPointName] で、サポートアクセスポイント名を入力します。

> **Note**
>
> これは前のステップで選択した Amazon S3 バケットに関連付けられたアクセスポイントです。Amazon S3 バケットに関連付けられているアクセスポイントがない場合は、[CreateNewSupportingAccessPoint] で [true] を選択すると、テンプレートを作成するようにテンプレートを構成できます。

9. [Next] を選択します。
10. [スタックオプションの設定] ページで、[次へ] をクリックします。

    このページのオプション設定の詳細については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation スタックオプションの設定」を参照してください。
11. Review ページで、スタックの作成を選択します。

AWS Cloud Development Kit (AWS CDK) の使用

AWS CDK を使用した Object Lambda アクセスポイントの設定の詳細については、AWS Cloud Development Kit (AWS CDK) API リファレンスの「AWS::S3ObjectLambda Construct Library」を参照してください。

CloudFormation テンプレートを使用して S3 Object Lambda のセットアップを自動化する

AWS CloudFormation テンプレートを使用すると、Amazon S3 Object Lambda アクセスポイントをすばやく作成できます。CloudFormation テンプレートは、関連するリソースを自動的に作成し、AWS Identity and Access Management (IAM) ロールを設定し、Object Lambda アクセスポイントを介してリクエストを自動的に処理する AWS Lambda 関数を設定します。CloudFormation テンプレートでベストプラクティスを実装し、セキュリティ体制を改善し、マニュアルプロセスによって引き起こされるエラーを減らすことができます。

この GitHub リポジトリには、CloudFormation テンプレートと Lambda 関数のソースコードが含まれています。テンプレートの使用方法については、「the section called “Object Lambda アクセスポイントの作成”」を参照してください。

テンプレートで提供される Lambda 関数が変換を実行することはありません。代わりに、S3 バケットからオブジェクトをそのまま返します。関数をクローン化し、独自の変換コードを追加して、データがアプリケーションに返されるときに変更および処理できます。関数の変更の詳細については、「the section called “Lambda 関数の変更”」と「the section called “Lambda 関数の記述”」を参照してください。

テンプレートの変更

新しいサポートアクセスポイントの作成

S3 Object Lambda は、Object Lambda アクセスポイントと、サポートアクセスポイントと呼ばれる、標準の S3 アクセスポイントの 2 つのアクセスポイントを使用します。Object Lambda アクセスポイントにリクエストを行うと、S3 は S3 Object Lambda の設定に応じて、ユーザーに代わって Lambda を呼び出すか、サポートアクセスポイントにリクエストを委任します。テンプレートのデプロイ時に、`aws cloudformation deploy` コマンドの一部として次のパラメータを渡すことにより、新しいサポートアクセスポイントを作成できます。

```
CreateNewSupportingAccessPoint=true
```

関数ペイロードの設定。

テンプレートをデプロイするときに `aws cloudformation deploy` コマンドの一部として、次のパラメータを渡すことにより、Lambda 関数に補足データを提供するようにペイロードを設定できます。

```
LambdaFunctionPayload="format=json"
```

Amazon CloudWatch モニタリングを有効にします。

テンプレートをデプロイするときに `aws cloudformation deploy` コマンドの一部として<次のパラメータを渡すことで、CloudWatch モニタリングを有効にできます。

```
EnableCloudWatchMonitoring=true
```

このパラメータにより、Amazon S3 リクエストメトリクスの Object Lambda アクセスポイントが有効になり、クライアント側とサーバー側のエラーをモニタリングする 2 つの CloudWatch アラームが作成されます。

> **Note**
>
> Amazon CloudWatch の使用には追加コストが発生します。Amazon S3 リクエストメトリクスの詳細については、「アクセスポイントのモニタリングとログ記録」を参照してください。
> 料金表については、「CloudWatch 料金表」を参照してください。

プロビジョニング済み同時実行の設定

レイテンシーを低減するために、Object Lambda アクセスポイントをサポートする Lambda 関数にプロビジョニングされた同時実行を設定するには、テンプレートを編集して、`Resources` の以下の行を含めます。

```
LambdaFunctionVersion:
      Type: AWS::Lambda::Version
      Properties:
        FunctionName: !Ref LambdaFunction
        ProvisionedConcurrencyConfig:
```

```
            ProvisionedConcurrentExecutions: Integer
```

> **Note**
>
> 同時実行のプロビジョニングには追加料金が発生します。プロビジョニングされた同時実行の詳細については、「AWS Lambdaデベロッパーガイド」の「Lambda プロビジョニング済み同時実行数の管理」を参照してください。
> 料金の詳細については、「AWS Lambda の料金表」を参照してください。

Lambda 関数の変更

`GetObject` リクエストのヘッダー値の変更

デフォルトでは、Lambda 関数は `Content-Length` と `ETag` を除くすべてのヘッダーを、署名付き URL リクエストから `GetObject` クライアントに転送します。Lambda 関数の変換コードに基づいて、新しいヘッダー値を `GetObject` クライアントに送信を選択できます。

Lambda 関数を更新して、新しいヘッダー値を `WriteGetObjectResponse` API オペレーションで渡すことで送信できます。

例えば、Lambda 関数が Amazon S3 オブジェクトのテキストを別の言語に変換する場合、`Content-Language` ヘッダーに新しい値を渡すことができます。これを行うには、以下のように `writeResponse` 関数を変更します。

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest
    },
    ...headers,
    ContentLanguage: 'my-new-language'
  }).promise();
```

```
}
```

サポートされているヘッダーの完全なリストについては、Amazon Simple Storage Service API リファレンスの WriteGetObjectResponse を参照してください。

メタデータヘッダーを返します。

Lambda 関数を更新して、新しいヘッダー値を WriteGetObjectResponse API オペレーションリクエストで渡すことで送信できます。

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest,
      'my-new-header': 'my-new-value'
    },
    ...headers
  }).promise();
}
```

新しいステータスコードを返します。

カスタムステータスコードを WriteGetObjectResponse API オペレーションリクエストで渡すことにより、GetObject クライアントに返すことができます。

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
```

```
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest
    },
    ...headers,
    StatusCode: Integer
  }).promise();
}
```

サポートされているステータスコードの完全なリストについては、Amazon Simple Storage Service API リファレンスの [WriteGetObjectResponse](#) を参照してください。

ソースオブジェクトに **Range** と **partNumber** パラメータを適用します。

デフォルトでは、CloudFormation テンプレートによって作成された Object Lambda アクセスポイントは、Range と partNumber のパラメータを処理できます。Lambda 関数は、要求された範囲または部品番号を変換されたオブジェクトに適用します。これを行うには、関数がオブジェクト全体をダウンロードして変換を実行する必要があります。場合によっては、変換されたオブジェクト範囲がソースオブジェクトの範囲に正確にマップされることがあります。つまり、ソースオブジェクトでバイト範囲 A–B を要求して変換を実行すると、オブジェクト全体を要求して変換を実行し、変換されたオブジェクトでバイト範囲 A–B を返すのと同じ結果が得られる可能性があります。

このような場合、Lambda 関数の実装を変更して、範囲または部品番号をソースオブジェクトに直接適用できます。このアプローチにより、関数全体のレイテンシーと必要なメモリが軽減されます。詳細については、「the section called “Range および partNumber ヘッダーの操作”」を参照してください。

Range と **partNumber** 処理の無効化

デフォルトでは、CloudFormation テンプレートによって作成された Object Lambda アクセスポイントは、Range と partNumber のパラメータを処理できます。この動作が不要な場合は、テンプレートから次の行を削除して無効化することができます。

```
AllowedFeatures:
  - GetObject-Range
  - GetObject-PartNumber
  - HeadObject-Range
  - HeadObject-PartNumber
```

大きなオブジェクトの変換

デフォルトでは、Lambda 関数は、S3 Object Lambda への応答のストリーミングをスタートする前に、メモリ内のオブジェクト全体を処理します。関数を変更して、変換の実行時に応答をストリーミングできます。これにより、変換のレイテンシーと Lambda 関数のメモリサイズを軽減できます。実装例については、ストリーミング圧縮コンテンツの例を参照してください。

Amazon S3 Object Lambda アクセスポイントの使用

Amazon S3 Object Lambda アクセスポイントを介してリクエストを行うことは、他のアクセスポイントを介してリクエストを行う場合と同様に機能します。アクセスポイントを介してリクエストを行う方法の詳細については、「アクセスポイントの使用」を参照してください。Amazon S3 コンソール、AWS Command Line Interface、(AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して Object Lambda アクセスポイントを介し、リクエストを行うことができます。

> **Important**
>
> Object Lambda アクセスポイントの Amazon リソースネーム (ARN) では、`s3-object-lambda` のサービス名が使用されます。したがって、Object Lambda アクセスポイントの ARN は、`arn:aws:s3` の代わりに `arn:aws:s3-object-lambda` で始まり、他のアクセスポイントと共に使用されます。

Object Lambda アクセスポイントの ARN を見つける方法

AWS CLI または AWS SDK で Object Lambda アクセスポイントを使用するには、Object Lambda アクセスポイントの Amazon リソースネーム (ARN) を知る必要があります。以下の例では、Amazon S3 コンソールまたは AWS CLI を使用して Object Lambda アクセスポイントの ARN を検索する方法を示します。

S3 コンソールの使用

コンソールを使用して Object Lambda アクセスポイントの ARN を見つける方法

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Object Lambda アクセスポイント] を選択します。
3. ARN をコピーする Object Lambda アクセスポイントの横にあるオプションボタンを選択します。
4. [ARN のコピー] を選択します。

AWS CLI の使用

AWS CLI を使用して Object Lambda アクセスポイントの ARN を見つける方法

1. AWS アカウント に関連付けられている Object Lambda アクセスポイントのリストを取得するにはで、以下のコマンドを実行します。コマンドを実行する前に、アカウント ID *111122223333* と AWS アカウント ID を交換してください。

```
aws s3control list-access-points-for-object-lambda --account-id 111122223333
```

2. コマンド出力を確認して、使用するオブジェクト Lambda アクセスポイント ARN を見つけます。先ほどのコマンドの出力は、次の例のようになります。

```
{
    "ObjectLambdaAccessPointList": [
        {
            "Name": "my-object-lambda-ap",
            "ObjectLambdaAccessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/my-object-lambda-ap"
        },
        ...
    ]
}
```

S3 バケット Object Lambda アクセスポイントでのバケット形式のエイリアスの使用方法

Object Lambda アクセスポイントを作成すると、Amazon S3 は Object Lambda アクセスポイントの固有のエイリアスを自動的に生成します。アクセスポイントのデータプレーンオペレーションのリクエストで、Amazon S3 バケット名や Object Lambda アクセスポイント Amazon Resource Name (ARN) の代わりに、このエイリアスを使用することができます。これらのオペレーションのリストについては、「AWS サービスとアクセスポイントの互換性」を参照してください。

Object Lambda アクセスポイントのエイリアス名は、Amazon S3 バケットと同じ名前空間内に作成されます。このエイリアス名は自動的に生成され、変更できません。既存の Object Lambda アクセスポイントには、エイリアスが自動的に割り当てられて使用されます。Object Lambda アクセスポイントのエイリアス名は、有効な Amazon S3 バケット名のすべての要件を満たしており、次の部分で構成されています。

Object Lambda Access Point name prefix-*metadata*--ol-s3

> **Note**
>
> `--ol-s3` サフィックスは、Object Lambda アクセスポイントのエイリアス名用に予約されており、バケット名や Object Lambda アクセスポイント名には使用できません。Amazon S3 バケット命名規則の詳細については、「バケットの名前付け」を参照してください。

以下は、*my-object-lambda-access-point* という名前の Object Lambda アクセスポイントの ARN と Object Lambda アクセスポイントのエイリアスの例です。

- ARN — `arn:aws:s3-object-lambda:`*region*`:`*account-id*`:accesspoint/`*my-object-lambda-access-point*
- Object Lambda アクセスポイントのエイリアス — *my-object-lambda-acc-1a4n8yjrb3kda96f67zwrwiiuse1a*`--ol-s3`

Object Lambda アクセスポイントを使用する場合、コードを大幅に変更しなくても Object Lambda アクセスポイントのエイリアス名を使用できます。

Object Lambda アクセスポイントを削除すると、Object Lambda アクセスポイントのエイリアス名は非アクティブになり、プロビジョニングも解除されます。

Object Lambda アクセスポイントのエイリアスを見つける方法

S3 コンソールの使用

コンソールを使用して Object Lambda アクセスポイントのエイリアスを見つけるには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Object Lambda アクセスポイント] を選択します。
3. 使用する Object Lambda アクセスポイントについて、Object Lambda アクセスポイントのエイリアス値をコピーします。

AWS CLI の使用

Object Lambda アクセスポイントを作成すると、次のコマンド例に示すように、Amazon S3 によって Object Lambda アクセスポイントのエイリアス名が自動的に生成されます。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。AWS CLI を使用

して Object Lambda アクセスポイントを作成する方法については、「AWS CLI を使用して Object Lambda アクセスポイントを作成するには」を参照してください。

```
aws s3control create-access-point-for-object-lambda --account-id 111122223333 --
name my-object-lambda-access-point --configuration file://my-olap-configuration.json
{
    "ObjectLambdaAccessPointArn": "arn:aws:s3:region:111122223333:accesspoint/my-
access-point",
    "Alias": {
        "Value": "my-object-lambda-acc-1a4n8yjrb3kda96f67zwrwiiuse1a--ol-s3",
        "Status": "READY"
    }
}
```

生成された Object Lambda アクセスポイントのエイリアス名には 2 つのフィールドがあります。

- `Value` フィールドは、Obda Lambda アクセスポイントのエイリアス値です。
- `Status` フィールドは、Object Lambda アクセスポイントのエイリアスのステータスです。ステータスが `PROVISIONING` の場合、Amazon S3 は Object Lambda Access Point エイリアスをプロビジョニング中ですが、エイリアスはまだ使用できる状態ではありません。ステータスが `READY` の場合、Object Lambda Access Point エイリアスは正常にプロビジョニングされ、使用できる状態になっています。

REST API `ObjectLambdaAccessPointAlias` データ型の詳細については、Amazon Simple Storage Service API リファレンス の「CreateAccessPointForObjectLambda」および「ObjectLambdaAccessPointAlias」を参照してください。

Object Lambda アクセスポイントのエイリアスの使用方法

AWS サービスとアクセスポイントの互換性 に示されているオペレーションには、Amazon S3 バケット名の代わりに Object Lambda Access Point エイリアスを使用できます。

`get-bucket-location` コマンドの次の AWS CLI 例は、バケットのアクセスポイントエイリアスを使用して、バケットが置かれている AWS リージョン を返します。このコマンドを実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3api get-bucket-location --bucket my-object-lambda-
acc-w7i37nq6xuzgax3jw3oqtifiusw2a--ol-s3
```

```
{
    "LocationConstraint": "us-west-2"
}
```

リクエスト内の Object Lambda アクセスポイントのエイリアスが有効でない場合、エラーコード `InvalidAccessPointAliasError` が返されます。`InvalidAccessPointAliasError` の詳細については、Amazon Simple Storage Service API リファレンスの「List of Error Codes」(エラーコードのリスト) を参照してください。

Object Lambda アクセスポイントエイリアスの制限は、アクセスポイントエイリアスの制限と同じです。アクセスポイントエイリアスの制限の詳細については、「制限事項」を参照してください。

S3 Object Lambda アクセスポイントのセキュリティに関する考慮事項

Amazon S3 Object Lambda では、コンピューティングプラットフォームとして AWS Lambda のスケールと柔軟性を利用して、Amazon S3 を離れたデータに対してカスタム変換を実行できます。S3 と Lambda はデフォルトで安全なままですが、この安全性を維持するためには、Lambda 関数の作成者による特別な考慮が必要です。S3 Object Lambda では、すべてのアクセスが、認証されたプリンシパル (匿名アクセスなし) により、HTTPS 経由で行われる必要があります。

セキュリティリスクを軽減するために、以下を推奨します。

- Lambda 実行ロールの範囲を、可能な限り最小の権限セットに設定します。
- 可能な限り、提供された署名付き URL を通じて Lambda 関数が Amazon S3 にアクセスするようにしてください。

IAM ポリシーの設定

S3 アクセスポイントは AWS Identity and Access Management (IAM) リソースポリシーをサポートしています。これにより、リソース、ユーザー、その他の条件別にアクセスポイントの使用を制御できます。詳細については、「Object Lambda アクセスポイントの IAM ポリシーの設定」を参照してください。

暗号化動作

Object Lambda アクセスポイントは Amazon S3 と AWS Lambda の両方を使用するため、暗号化の動作には違いがあります。デフォルトの S3 暗号化の動作の詳細については、Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定 を参照してください。

- Object Lambda アクセスポイントで S3 サーバー側の暗号化を使用している場合、オブジェクトは Lambda に送信される前に復号化されます。オブジェクトが Lambda に送信されると、暗号化されずに処理されます (GET または HEAD リクエストの場合)。
- 暗号化キーがログに記録されないようにするため、S3 は、お客様が指定したキーを使用したサーバー側の暗号化 (SSE-C) で暗号化されたオブジェクトに対する GET および HEAD リクエストを拒否します。ただし、Lambda 関数は、クライアントが指定したキーにアクセスできる場合、これらのオブジェクトを取得する可能性があります。
- Object Lambda アクセスポイントで S3 クライアント側の暗号化を使用する際は、Lambda がオブジェクトを復号および再暗号化するための暗号化キーにアクセスできることを確認してください。

アクセスポイントセキュリティ

S3 Object Lambda は、Object Lambda アクセスポイントと、サポートアクセスポイントと呼ばれる、標準の S3 アクセスポイントの 2 つのアクセスポイントを使用します。Object Lambda アクセスポイントにリクエストを行うと、S3 は S3 Object Lambda の設定に応じて、ユーザーに代わって Lambda を呼び出すか、サポートアクセスポイントにリクエストを委任します。Lambda がリクエストに対して呼び出されると、S3 は、サポートアクセスポイントを通じて、ユーザーに代わってオブジェクトへの署名付き URL を生成します。Lambda 関数は、関数が呼び出されたときにこの URL を入力として受け取ります。

S3 を直接呼び出すのではなく、この署名付き URL を使用して元のオブジェクトを取得するように Lambda 関数を設定できます。このモデルを使用することで、より優れたセキュリティ境界をオブジェクトに適用できます。S3 バケットまたは S3 アクセスポイント経由のオブジェクト直接アクセスを、限定された IAM ロールまたはユーザーのセットに制限できます。このアプローチにより、Lambda 関数が「混乱した代理問題」のターゲットになることを防げます。これは、呼び出し元とは異なるアクセス許可を持つ誤って設定された関数が、本来は許可または拒否すべきでないオブジェクトへのアクセスを許可または拒否する可能性があるというものです。

Object Lambda アクセスポイントのパブリックアクセス

S3 Object Lambda は匿名またはパブリックアクセスを許可しません。これは、Amazon S3 が S3 Object Lambda リクエストを完了するために ID を承認する必要があるためです。Object Lambda アクセスポイントを介してリクエストを呼び出す際には、設定された Lambda 関数の `lambda:InvokeFunction` 許可が必要となります。同様に、Object Lambda アクセスポイントを介して他の API オペレーションを呼び出す際には、必須の `s3:*` 許可が必要となります。

これらのアクセス許可がないと、Lambda の呼び出しまたは S3 への委任のリクエストは HTTP 403 (Forbidden) エラーとして失敗します。すべてのアクセスは、認証されたプリンシパルによって行わ

れる必要があります。パブリックアクセスが必要な場合は、Lambda@Edge を代替として使用できます。詳細については、Amazon CloudFront 開発者ガイドの Lambda@Edge を使用したエッジでのカスタマイズを参照してください。

Object Lambda アクセスポイント IP アドレス

`describe-managed-prefix-lists` サブネットはゲートウェイ仮想プライベートクラウド (VPC) エンドポイントをサポートしており、VPC エンドポイントのルーティングテーブルに関連しています。Object Lambda アクセスポイントはゲートウェイ VPC をサポートしていないため、その IP 範囲は欠落しています。欠落している範囲は Amazon S3 に属しますが、ゲートウェイ VPC エンドポイントではサポートされていません。`describe-managed-prefix-lists` の詳細については、「Amazon EC2 API リファレンス」の「DescribeManagedPrefixLists」と「AWS 全般のリファレンス」の「AWS IP アドレスの範囲」を参照してください。

Object Lambda アクセスポイントの IAM ポリシーの設定

Amazon S3 アクセスポイントは AWS Identity and Access Management (IAM) リソースポリシーをサポートしています。これにより、リソース、ユーザー、またはその他の条件別にアクセスポイントの使用を制御できます。Object Lambda アクセスポイントのオプションリソースポリシー、またはサポートアクセスポイントのリソースポリシーを使用してアクセスを制御できます。詳細な例については、チュートリアル: S3 Object Lambda を使用したアプリケーションのデータの変換 および チュートリアル: S3 Object Lambda と Amazon Comprehend を使用した PII データの検出と編集 を参照してください。

次の 4 つのリソースに Object Lambda アクセスポイントを使用するための許可が付与されている必要があります。

- ユーザーやロールなどの IAM ID。IAM ID およびベストプラクティスの詳細については、IAM ユーザーガイドの「IAM ID (ユーザー、ユーザーグループ、ロール)」を参照してください。
- バケットおよび関連する標準アクセスポイント。Object Lambda アクセスポイントを使用する場合、この標準アクセスポイントはサポートアクセスポイントと呼ばれます。
- Object Lambda アクセスポイント。
- AWS Lambda 関数。

> **Important**
>
> ポリシーを保存する前に、AWS Identity and Access Management Access Analyzer からのセキュリティ警告、エラー、一般的な警告、および提案を解決してください。IAM Access

Analyzer は、IAM ポリシーの文法とベストプラクティスに照らしてポリシーチェックを行います。これらのチェックにより、機能的でセキュリティのベストプラクティスに準拠したポリシーを作成するのに、役立つ結果と実行可能なレコメンデーションが示されます。IAM Access Analyzer を使用したポリシーの検証の詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer のポリシーの検証」を参照してください。IAM Access Analyzer によって返される警告、エラー、および提案のリストを表示するには、「IAM Access Analyzer ポリシーチェックリファレンス」を参照してください。

以下のポリシーは、次のリソースがあることを前提としています。

- 次の Amazon リソースネーム (ARN) を持つ Amazon S3 バケット:

  `arn:aws:s3:::DOC-EXAMPLE-BUCKET1`
- このバケット上の Amazon S3 Standard アクセスポイントに次の ARN があること。

  `arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point`
- Object Lambda アクセスポイントに次の ARN があること。

  `arn:aws:s3-object-lambda:us-east-1:111122223333:accesspoint/my-object-lambda-ap`
- AWS Lambda 関数に次の ARN があること。

  `arn:aws:lambda:us-east-1:111122223333:function:MyObjectLambdaFunction`

Note

アカウントから Lambda 関数を使用する場合は、ポリシーステートメントに特定の関数のバージョンを含める必要があります。次の ARN の例では、バージョンは `1` で示されています。
`arn:aws:lambda:us-east-1:111122223333:function:MyObjectLambdaFunction:1`
Lambda は、バージョン $LATEST への IAM ポリシーの追加をサポートしていません。Lambda 関数のバージョンの詳細については、「AWS Lambda 開発者ガイド」の「Lambda 関数のバージョン」を参照してください。

Example - アクセスコントロールを標準アクセスポイントに委任するバケットポリシー

次の S3 バケットポリシーの例は、バケットのアクセスコントロールをバケットの標準アクセスポイントに委任します。このポリシーでは、バケット所有者のアカウントが所有するすべてのアクセスポイントへのフルアクセスを許可しています。したがって、このバケットへのすべてのアクセスは、そのアクセスポイントにアタッチされているポリシーによってコントロールされます。ユーザーはアクセスポイントを介してのみバケットから読み取ることができます。つまり、オペレーションは、アクセスポイントを介してのみ呼び出すことができます。詳細については、「アクセスポイントへのアクセスコントロールの委任」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS":"account-ARN"},
        "Action" : "*",
        "Resource" : [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET1",
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
        ],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Bucket owner's account
 ID" }
        }
    }]
}
```

Example — Object Lambda アクセスポイントを使用するために必要なアクセス許可をユーザーに付与する IAM ポリシー

次の IAM ポリシーは、Lambda 関数、標準アクセスポイント、および Object Lambda アクセスポイントに対するアクセス許可をユーザーに付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowLambdaInvocation",
      "Action": [
        "lambda:InvokeFunction"
      ],
```

```
      "Effect": "Allow",
      "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:MyObjectLambdaFunction:1",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "s3-object-lambda.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AllowStandardAccessPointAccess",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "s3-object-lambda.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AllowObjectLambdaAccess",
      "Action": [
        "s3-object-lambda:Get*",
        "s3-object-lambda:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3-object-lambda:us-east-1:111122223333:accesspoint/my-
object-lambda-ap"
    }
  ]
}
```

Lambda 実行ロールの許可を有効化する

Object Lambda アクセスポイントに GET リクエストが行われた場合、Lambda 関数には、S3 Object Lambda アクセスポイントにデータを送信する許可が必要です。このアクセス許可は、Lambda 関数の実行ロールで `s3-object-lambda:WriteGetObjectResponse` 許可を有効にすることで提供されます。新しい実行ロールを使作成することも、既存のロールを更新することもできます。

> **Note**
>
> 関数に `s3-object-lambda:WriteGetObjectResponse` 許可が必要なのは、GET リクエストを行う場合のみです。

IAM コンソールで実行ロールを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。
3. [ロールの作成] を選択します。
4. [一般的なユースケース] で、[Lambda] を選択します。
5. [Next] を選択します。
6. [Add permissions] (許可の追加) ページで、AWS マネージドポリシー `AmazonS3ObjectLambdaExecutionRolePolicy` を検索し、ポリシー名の横にあるチェックボックスをオンにします。

   このポリシーには `s3-object-lambda:WriteGetObjectResponse` アクションを含める必要があります。
7. [Next] を選択します。
8. [Name, review, and create] (名前、確認、作成) ページで、[Role name] (ロール名) に **`s3-object-lambda-role`** と入力します。
9. (オプション) このロールの説明とタグを追加します。
10. [ロールの作成] を選択します。
11. 新しく作成した **`s3-object-lambda-role`** を Lambda 関数の実行ロールとして適用します。これは、Lambda コンソールで Lambda 関数の作成中または作成後に実行できます。

実行ロールの詳細については、「AWS Lambda デベロッパーガイド」の「Lambda 実行ロール」を参照してください。

Object Lambda アクセスポイントでのコンテキストキーの使用

S3 Object Lambda は、リクエストの接続や署名に関連する `s3-object-lambda:TlsVersion` や `s3-object-lambda:AuthType` などのコンテキストキーを評価します。`s3:prefix` など、他のすべてのコンテキストキーは、Amazon S3 によって評価されます。

Object Lambda アクセスポイントの CORS サポート

S3 Object Lambda がブラウザからリクエストを受信するか、リクエストに `Origin` ヘッダーが含まれている場合、S3 Object Lambda は常に `“AllowedOrigins":"*"` ヘッダーフィールドを追加します。

詳細については、「Cross-Origin Resource Sharing (CORS) の使用」を参照してください。

S3 Object Lambda アクセスポイントの Lambda 関数の記述

このセクションでは、Amazon S3 Object Lambda アクセスポイントで使用するための AWS Lambda 関数を記述する方法について詳述します。

S3 Object Lambda の一部のタスクにおける完全なエンドツーエンド手順については、以下を参照してください。

- チュートリアル: S3 Object Lambda を使用したアプリケーションのデータの変換
- チュートリアル: S3 Object Lambda と Amazon Comprehend を使用した PII データの検出と編集
- チュートリアル: S3 Object Lambda を使用して、取得時に画像に動的に透かしを入れる

トピック

- Lambda での GetObject リクエストの使用
- Lambda での HeadObject リクエストの使用
- Lambda での ListObjects リクエストの使用
- Lambda での ListObjectsV2 リクエストの使用
- イベントコンテキストの形式と使用法
- Range および partNumber ヘッダーの操作

Lambda での `GetObject` リクエストの使用

このセクションでは、Object Lambda アクセスポイントが `GetObject` 用に Lambda 関数を呼び出すように設定されていることを前提としています。S3 Object Lambda には、Amazon S3 API オペ

レーション、WriteGetObjectResponse が含まれています。これにより、Lambda 関数は、カスタマイズされたデータとレスポンスヘッダーを GetObject 発信者に提供できるようになります。

WriteGetObjectResponse は、処理のニーズに基づいて、ステータスコード、レスポンスヘッダー、レスポンス本文を広範囲に制御できます。WriteGetObjectResponse を使用すると、変換されたオブジェクト全体、変換されたオブジェクトの一部、またはアプリケーションのコンテキストに基づくその他のレスポンスに対して応答することができます。次のセクションでは、WriteGetObjectResponse API オペレーションを使用した一意の例を示します。

- 例 1: HTTP ステータスコード 403 (Forbidden) で応答します。
- 例 2: 変換された画像で応答する
- 例 3: 圧縮されたコンテンツをストリーミングする

例 1: HTTP ステータスコード 403 (Forbidden) で応答します。

WriteGetObjectResponse を使用して、オブジェクトの内容に基づいて HTTP ステータスコード 403 (Forbidden) で応答できます。

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import java.io.ByteArrayInputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example1 {

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();
```

```
        // Check to see if the request contains all of the necessary information.
        // If it does not, send a 4XX response and a custom error code and message.
        // Otherwise, retrieve the object from S3 and stream it
        // to the client unchanged.
        var tokenIsNotPresent = !
event.getUserRequest().getHeaders().containsKey("requiredToken");
        if (tokenIsNotPresent) {
            s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                    .withRequestRoute(event.outputRoute())
                    .withRequestToken(event.outputToken())
                    .withStatusCode(403)
                    .withContentLength(0L).withInputStream(new
 ByteArrayInputStream(new byte[0]))
                    .withErrorCode("MissingRequiredToken")
                    .withErrorMessage("The required token was not present in the
 request."));
            return;
        }

        // Prepare the presigned URL for use and make the request to S3.
        HttpClient httpClient = HttpClient.newBuilder().build();
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // Stream the original bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(presignedResponse.body()));
    }
}
```

Python

```
import boto3
import requests

def handler(event, context):
    s3 = boto3.client('s3')

    """
```

```
   Retrieve the operation context object from the event. This object indicates
where the WriteGetObjectResponse request
   should be delivered and contains a presigned URL in 'inputS3Url' where we can
download the requested object from.
   The 'userRequest' object has information related to the user who made this
'GetObject' request to
   S3 Object Lambda.
   """
   get_context = event["getObjectContext"]
   user_request_headers = event["userRequest"]["headers"]

   route = get_context["outputRoute"]
   token = get_context["outputToken"]
   s3_url = get_context["inputS3Url"]

   # Check for the presence of a 'CustomHeader' header and deny or allow based on
that header.
   is_token_present = "SuperSecretToken" in user_request_headers

   if is_token_present:
       # If the user presented our custom 'SuperSecretToken' header, we send the
requested object back to the user.
       response = requests.get(s3_url)
       s3.write_get_object_response(RequestRoute=route, RequestToken=token,
Body=response.content)
   else:
       # If the token is not present, we send an error back to the user.
       s3.write_get_object_response(RequestRoute=route, RequestToken=token,
StatusCode=403,
       ErrorCode="NoSuperSecretTokenFound", ErrorMessage="The request was not
secret enough.")

   # Gracefully exit the Lambda function.
   return { 'status_code': 200 }
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;

exports.handler = async (event) => {
    const s3 = new S3();
```

```
    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and contains a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    // The 'userRequest' object has information related to the user who made this
 'GetObject' request to S3 Object Lambda.
    const { userRequest, getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // Check for the presence of a 'CustomHeader' header and deny or allow based on
 that header.
    const isTokenPresent = Object
        .keys(userRequest.headers)
        .includes("SuperSecretToken");

    if (!isTokenPresent) {
        // If the token is not present, we send an error back to the user. The
 'await' in front of the request
        // indicates that we want to wait for this request to finish sending before
 moving on.
        await s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            StatusCode: 403,
            ErrorCode: "NoSuperSecretTokenFound",
            ErrorMessage: "The request was not secret enough.",
        }).promise();
    } else {
        // If the user presented our custom 'SuperSecretToken' header, we send the
 requested object back to the user.
        // Again, note the presence of 'await'.
        const presignedResponse = await axios.get(inputS3Url);
        await s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            Body: presignedResponse.data,
        }).promise();
    }

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

例 2: 変換された画像で応答する

画像変換を実行する場合、ソースオブジェクトの処理を開始する前に、ソースオブジェクトのすべてのバイトが必要になる可能性があります。この場合、WriteGetObjectResponse リクエストは、オブジェクト全体を 1 回の呼び出しでリクエスト元のアプリケーションに返します。

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.awt.Image;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example2 {

    private static final int HEIGHT = 250;
    private static final int WIDTH = 250;

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();
        HttpClient httpClient = HttpClient.newBuilder().build();

        // Prepare the presigned URL for use and make the request to S3.
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // The entire image is loaded into memory here so that we can resize it.
        // Once the resizing is completed, we write the bytes into the body
```

```
        // of the WriteGetObjectResponse request.
        var originalImage = ImageIO.read(presignedResponse.body());
        var resizingImage = originalImage.getScaledInstance(WIDTH, HEIGHT,
 Image.SCALE_DEFAULT);
        var resizedImage = new BufferedImage(WIDTH, HEIGHT,
 BufferedImage.TYPE_INT_RGB);
        resizedImage.createGraphics().drawImage(resizingImage, 0, 0, WIDTH, HEIGHT,
 null);

        var baos = new ByteArrayOutputStream();
        ImageIO.write(resizedImage, "png", baos);

        // Stream the bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(new ByteArrayInputStream(baos.toByteArray())));
    }
}
```

Python

```
import boto3
import requests
import io
from PIL import Image

def handler(event, context):
    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to
    S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    """
```

```
    In this case, we're resizing .png images that are stored in S3 and are
 accessible through the presigned URL
    'inputS3Url'.
    """
    image_request = requests.get(s3_url)
    image = Image.open(io.BytesIO(image_request.content))
    image.thumbnail((256,256), Image.ANTIALIAS)

    transformed = io.BytesIO()
    image.save(transformed, "png")

    # Send the resized image back to the client.
    s3 = boto3.client('s3')
    s3.write_get_object_response(Body=transformed.getvalue(), RequestRoute=route,
 RequestToken=token)

    # Gracefully exit the Lambda function.
    return { 'status_code': 200 }
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;
const sharp = require('sharp');

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    const { getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // In this case, we're resizing .png images that are stored in S3 and are
 accessible through the presigned URL
    // 'inputS3Url'.
    const { data } = await axios.get(inputS3Url, { responseType: 'arraybuffer' });

    // Resize the image.
    const resized = await sharp(data)
        .resize({ width: 256, height: 256 })
```

```
        .toBuffer();

    // Send the resized image back to the client.
    await s3.writeGetObjectResponse({
        RequestRoute: outputRoute,
        RequestToken: outputToken,
        Body: resized,
    }).promise();

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

例 3: 圧縮されたコンテンツをストリーミングする

オブジェクトを圧縮すると、圧縮データは増分的に生成されます。したがって、準備ができたらすぐに圧縮されたデータを返すために、WriteGetObjectResponse リクエストを使用できます。この例で示すように、完了した変換の長さを知る必要はありません。

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example3 {

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();
        HttpClient httpClient = HttpClient.newBuilder().build();

        // Request the original object from S3.
```

```
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // Consume the incoming response body from the presigned request,
        // apply our transformation on that data, and emit the transformed bytes
        // into the body of the WriteGetObjectResponse request as soon as they're
 ready.
        // This example compresses the data from S3, but any processing pertinent
        // to your application can be performed here.
        var bodyStream = new GZIPCompressingInputStream(presignedResponse.body());

        // Stream the bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(bodyStream));
    }

}
```

Python

```
import boto3
import requests
import zlib
from botocore.config import Config


"""
A helper class to work with content iterators. Takes an interator and compresses the
 bytes that come from it. It
implements 'read' and '__iter__' so that the SDK can stream the response.
"""
class Compress:
    def __init__(self, content_iter):
        self.content = content_iter
        self.compressed_obj = zlib.compressobj()

    def read(self, _size):
        for data in self.__iter__()
            return data
```

```
    def __iter__(self):
        while True:
            data = next(self.content)
            chunk = self.compressed_obj.compress(data)
            if not chunk:
                break

            yield chunk

        yield self.compressed_obj.flush()


def handler(event, context):
    """
    Setting the 'payload_signing_enabled' property to False allows us to send a
 streamed response back to the client.
    in this scenario, a streamed response means that the bytes are not buffered into
 memory as we're compressing them,
    but instead are sent straight to the user.
    """
    my_config = Config(
        region_name='eu-west-1',
        signature_version='s3v4',
        s3={
            "payload_signing_enabled": False
        }
    )
    s3 = boto3.client('s3', config=my_config)

    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    # Compress the 'get' request stream.
    with requests.get(s3_url, stream=True) as r:
```

```
        compressed = Compress(r.iter_content())

        # Send the stream back to the client.
        s3.write_get_object_response(Body=compressed, RequestRoute=route,
 RequestToken=token, ContentType="text/plain",
                                     ContentEncoding="gzip")

    # Gracefully exit the Lambda function.
    return {'status_code': 200}
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;
const zlib = require('zlib');

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    const { getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // Download the object from S3 and process it as a stream, because it might be a
 huge object and we don't want to
    // buffer it in memory. Note the use of 'await' because we want to wait for
 'writeGetObjectResponse' to finish
    // before we can exit the Lambda function.
    await axios({
        method: 'GET',
        url: inputS3Url,
        responseType: 'stream',
    }).then(
        // Gzip the stream.
        response => response.data.pipe(zlib.createGzip())
    ).then(
        // Finally send the gzip-ed stream back to the client.
        stream => s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
```

```
            Body: stream,
            ContentType: "text/plain",
            ContentEncoding: "gzip",
        }).promise()
    );

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

> **Note**
>
> S3 Object Lambda では、発信者に完全なレスポンスを送信するのに最大 60 秒かかります。`WriteGetObjectResponse` リクエストの場合、実際の利用可能な時間は短くなる可能性があります。例えば、Lambda 関数のタイムアウトが 60 秒未満である可能性があります。それ以外にも、発信者のタイムアウトがより厳しい場合があります。

元の発信者が HTTP ステータスコード 500 以外のレスポンス (内部サーバエラー) を受信するには、`WriteGetObjectResponse` コールが完了する必要があります。`WriteGetObjectResponse` API オペレーションが呼び出される前に、Lambda 関数が例外的に、またはその他の方法で返された場合、元の発信者は 500 (内部サーバーエラー) レスポンスを受け取ります。レスポンスが完了するまでの間にスローされた例外は、発信者へのレスポンスが切り捨てられます。Lambda 関数が `WriteGetObjectResponse` API コールから HTTP ステータスコード 200 (OK) レスポンスを受信した場合、元の発信者は完全なリクエストを送信しています。例外がスローされたかどうかにかかわらず、Lambda 関数のレスポンスは S3 Object Lambda では無視されます。

この `WriteGetObjectResponse` API オペレーションを呼び出すとき、Amazon S3 はイベントコンテキストからのルートトークンとリクエストトークンを必要とします。詳細については、「イベントコンテキストの形式と使用法」を参照してください。

ルートトークンとリクエストトークンのパラメータは、`WriteGetObjectResult` レスポンスを元の発信者に接続するために必要です。通常、500 (内部サーバーエラー) レスポンスを再試行するのが適切ですが、リクエストトークンはワンタイムトークンであることに注意してください。それ以降に使用を試みると、HTTP ステータスコード 400 (Bad Request) レスポンスが返されることになる可能性があります。ルートトークンとリクエストトークンを使用した `WriteGetObjectResponse` への呼び出しは、呼び出された Lambda 関数から作成する必要はありませんが、同じアカウント内のア

イデンティティによって作成する必要があります。この呼び出しは、Lambda 関数の実行を完了する前に完了する必要があります。

Lambda での **HeadObject** リクエストの使用

このセクションでは、Object Lambda アクセスポイントが HeadObject 用に Lambda 関数を呼び出すように設定されていることを前提としています。Lambda は headObjectContext というキーを含む JSON ペイロードを受け取ります。コンテキスト内には、inputS3Url というプロパティが 1 つあります。これは、HeadObject をサポートしているアクセスポイントの署名付き URL です。

署名付き URL に次のプロパティが指定されている場合、署名付き URL にはそれらが含まれます。

- versionId (クエリパラメータ内)
- requestPayer (x-amz-request-payer ヘッダー内)
- expectedBucketOwner (x-amz-expected-bucket-owner ヘッダー内)

他のプロパティは署名付きではないため、含まれません。ヘッダーとして送信される署名されていないオプションは、userRequest ヘッダーにある署名付き URL を呼び出すときに、手動でリクエストに追加できます。サーバー側の暗号化オプションは、HeadObject ではサポートされていません。

リクエスト構文 URI パラメータについては、「Amazon Simple Storage Service API リファレンス」の「HeadObject」を参照してください。

次の例は、HeadObject の Lambda JSON 入力ペイロードを示しています。

```
{
  "xAmzRequestId": "requestId",
  "**headObjectContext**": {
    "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/example?X-Amz-Security-Token=<snip>"
  },
  "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
  },
  "userRequest": {
```

```
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
             "attributes": {
             "mfaAuthenticated": "false",
             "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
        },
        "sessionIssuer": {
             "type": "Role",
             "principalId": "principalId",
             "arn": "arn:aws:iam::111122223333:role/Admin",
             "accountId": "111122223333",
             "userName": "Admin"
             }
        }
     },
   "protocolVersion": "1.00"
}
```

Lambda 関数は、HeadObject 呼び出しで返されるヘッダーと値を含む JSON オブジェクトを返します。

次の例は、HeadObject の Lambda レスポンス JSON の構造を示しています。

```
{
    "statusCode": <number>; // Required
    "errorCode": <string>;
    "errorMessage": <string>;
    "headers": {
        "Accept-Ranges": <string>,
```

```
        "x-amz-archive-status": <string>,
        "x-amz-server-side-encryption-bucket-key-enabled": <boolean>,
        "Cache-Control": <string>,
        "Content-Disposition": <string>,
        "Content-Encoding": <string>,
        "Content-Language": <string>,
        "Content-Length": <number>, // Required
        "Content-Type": <string>,
        "x-amz-delete-marker": <boolean>,
        "ETag": <string>,
        "Expires": <string>,
        "x-amz-expiration": <string>,
        "Last-Modified": <string>,
        "x-amz-missing-meta": <number>,
        "x-amz-object-lock-mode": <string>,
        "x-amz-object-lock-legal-hold": <string>,
        "x-amz-object-lock-retain-until-date": <string>,
        "x-amz-mp-parts-count": <number>,
        "x-amz-replication-status": <string>,
        "x-amz-request-charged": <string>,
        "x-amz-restore": <string>,
        "x-amz-server-side-encryption": <string>,
        "x-amz-server-side-encryption-customer-algorithm": <string>,
        "x-amz-server-side-encryption-aws-kms-key-id": <string>,
        "x-amz-server-side-encryption-customer-key-MD5": <string>,
        "x-amz-storage-class": <string>,
        "x-amz-tagging-count": <number>,
        "x-amz-version-id": <string>,
        <x-amz-meta-headers>: <string>, // user-defined metadata
        "x-amz-meta-meta1": <string>, // example of the user-defined metadata header,
 it will need the x-amz-meta prefix
        "x-amz-meta-meta2": <string>
        ...
    };
}
```

次の例は、JSON を返す前に必要に応じてヘッダー値を変更することにより、署名付き URL を使用してレスポンスを入力する方法を示しています。

Python

```
import requests
```

```
def lambda_handler(event, context):
    print(event)

    # Extract the presigned URL from the input.
    s3_url = event["headObjectContext"]["inputS3Url"]

    # Get the head of the object from S3.
    response = requests.head(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        return {
            "statusCode": response.status_code,
            "errorCode": "RequestFailure",
            "errorMessage": "Request to S3 failed"
    }

    # Store the headers in a dictionary.
    response_headers = dict(response.headers)

    # This obscures Content-Type in a transformation, it is optional to add
    response_headers["Content-Type"] = ""

    # Return the headers to S3 Object Lambda.
    return {
        "statusCode": response.status_code,
        "headers": response_headers
        }
```

Lambda での **ListObjects** リクエストの使用

このセクションでは、Object Lambda アクセスポイントが `ListObjects` 用に Lambda 関数を呼び出すように設定されていることを前提としています。Lambda は、`listObjectsContext` という新しいオブジェクトを持つ JSON ペイロードを受け取ります。`listObjectsContext` には、`inputS3Url` という単一のプロパティが含まれており、これは `ListObjects` をサポートしているアクセスポイント用の署名付き URL です。

`GetObject` や `HeadObject` とは異なり、署名付き URL には、次のプロパティが指定されている場合はそれらが含まれます。

- すべてのクエリパラメータ

- `requestPayer` (`x-amz-request-payer` ヘッダー内)
- `expectedBucketOwner` (`x-amz-expected-bucket-owner` ヘッダー内)

リクエスト構文 URI パラメータについては、「Amazon Simple Storage Service API リファレンス」の「ListObjects」を参照してください。

> **Important**
>
> アプリケーションを開発する場合は、新しいバージョンの ListObjectSv2 を使用することをお勧めします。Amazon S3 は、下位互換性のために、引き続き `ListObjects` をサポートしています。

次の例は、`ListObjects` の Lambda JSON 入力ペイロードを示しています。

```
{
    "xAmzRequestId": "requestId",
     "**listObjectsContext**": {
     "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/?X-Amz-Security-Token=<snip>",
     },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
```

```
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
  "protocolVersion": "1.00"
}
```

Lambda 関数は、S3 Object Lambda から返されるステータスコード、XML 結果のリスト、またはエラー情報を含む JSON オブジェクトを返します。

S3 Object Lambda は `listResultXml` の処理も検証もせず、代わりに `ListObjects` 呼び出し元に転送します。`listBucketResult` については、S3 Object Lambda は特定のプロパティが特定のタイプであることを想定し、解析できない場合は例外をスローします。 `listResultXml` および `listBucketResult` を同時に提供することはできません。

次の例は、署名付き URL を使用して Amazon S3 を呼び出し、その結果を使用してエラーチェックを含むレスポンスを入力する方法を示しています。

Python

```
import requests
import xmltodict

def lambda_handler(event, context):
    # Extract the presigned URL from the input.
    s3_url = event["listObjectsContext"]["inputS3Url"]


    # Get the head of the object from Amazon S3.
    response = requests.get(s3_url)
```

```
    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        error = xmltodict.parse(response.content)
        return {
            "statusCode": response.status_code,
            "errorCode": error["Error"]["Code"],
            "errorMessage": error["Error"]["Message"]
        }

    # Store the XML result in a dict.
    response_dict = xmltodict.parse(response.content)

    # This obscures StorageClass in a transformation, it is optional to add
    for item in response_dict['ListBucketResult']['Contents']:
        item['StorageClass'] = ""

    # Convert back to XML.
    listResultXml = xmltodict.unparse(response_dict)

    # Create response with listResultXml.
    response_with_list_result_xml = {
        'statusCode': 200,
        'listResultXml': listResultXml
    }

    # Create response with listBucketResult.
    response_dict['ListBucketResult'] =
 sanitize_response_dict(response_dict['ListBucketResult'])
    response_with_list_bucket_result = {
        'statusCode': 200,
        'listBucketResult': response_dict['ListBucketResult']
    }

    # Return the list to S3 Object Lambda.
    # Can return response_with_list_result_xml or response_with_list_bucket_result
    return response_with_list_result_xml

# Converting the response_dict's key to correct casing
def sanitize_response_dict(response_dict: dict):
    new_response_dict = dict()
    for key, value in response_dict.items():
        new_key = key[0].lower() + key[1:] if key != "ID" else 'id'
        if type(value) == list:
```

```
            newlist = []
            for element in value:
                if type(element) == type(dict()):
                    element = sanitize_response_dict(element)
                newlist.append(element)
            value = newlist
        elif type(value) == dict:
            value = sanitize_response_dict(value)
        new_response_dict[new_key] = value
    return new_response_dict
```

次の例は、ListObjects の Lambda レスポンス JSON の構造を示しています。

```
{
  "statusCode": <number>; // Required
  "errorCode": <string>;
  "errorMessage": <string>;
  "listResultXml": <string>; // This can also be Error XML string in case S3 returned
 error response when calling the pre-signed URL

  "listBucketResult": {  // listBucketResult can be provided instead of listResultXml,
 however they can not both be provided in the JSON response
       "name": <string>,  // Required for 'listBucketResult'
       "prefix": <string>,
       "marker": <string>,
       "nextMarker": <string>,
       "maxKeys": <int>,   // Required for 'listBucketResult'
       "delimiter": <string>,
       "encodingType": <string>
       "isTruncated": <boolean>,  // Required for 'listBucketResult'
       "contents": [  {
           "key": <string>,  // Required for 'content'
           "lastModified": <string>,
           "eTag": <string>,
           "checksumAlgorithm": <string>,   // CRC32,  CRC32C,  SHA1,  SHA256
           "size": <int>,   // Required for 'content'
           "owner": {
               "displayName": <string>,  // Required for 'owner'
               "id": <string>,  // Required for 'owner'
           },
           "storageClass": <string>
           },
```

```
        ...
        ],
        "commonPrefixes": [  {
            "prefix": <string>   // Required for 'commonPrefix'
        },
        ...
        ],
    }
}
```

Lambda での **ListObjectsV2** リクエストの使用

このセクションでは、Object Lambda アクセスポイントが ListObjectsV2 用に Lambda 関数を呼び出すように設定されていることを前提としています。Lambda は、listObjectsV2Context という新しいオブジェクトを持つ JSON ペイロードを受け取ります。listObjectsV2Context には、inputS3Url という単一のプロパティが含まれており、これは ListObjectsV2 をサポートしているアクセスポイント用の署名付き URL です。

GetObject や HeadObject とは異なり、署名付き URL には、次のプロパティが指定されている場合はそれらが含まれます。

- すべてのクエリパラメータ
- requestPayer (x-amz-request-payer ヘッダー内)
- expectedBucketOwner (x-amz-expected-bucket-owner ヘッダー内)

リクエスト構文 URI パラメータについては、「Amazon Simple Storage Service API リファレンス」の「ListObjectsV2」を参照してください。

次の例は、ListObjectsV2 の Lambda JSON 入力ペイロードを示しています。

```
{
    "xAmzRequestId": "requestId",
     "**listObjectsV2Context**": {
     "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/?list-type=2&X-Amz-Security-Token=<snip>",
     },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
```

```
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
  "protocolVersion": "1.00"
}
```

Lambda 関数は、S3 Object Lambda から返されるステータスコード、XML 結果のリスト、またはエラー情報を含む JSON オブジェクトを返します。

S3 Object Lambda は `listResultXml` の処理も検証もせず、代わりに `ListObjectsV2` 呼び出し元に転送します。`listBucketResult` については、S3 Object Lambda は特定のプロパティが特定のタイプであることを想定し、解析できない場合は例外をスローします。 `listResultXml` および `listBucketResult` を同時に提供することはできません。

次の例は、署名付き URL を使用して Amazon S3 を呼び出し、その結果を使用してエラーチェックを含むレスポンスを入力する方法を示しています。

Python

```
import requests
import xmltodict

def lambda_handler(event, context):
    # Extract the presigned URL from the input.
    s3_url = event["listObjectsV2Context"]["inputS3Url"]


    # Get the head of the object from Amazon S3.
    response = requests.get(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        error = xmltodict.parse(response.content)
        return {
            "statusCode": response.status_code,
            "errorCode": error["Error"]["Code"],
            "errorMessage": error["Error"]["Message"]
        }

    # Store the XML result in a dict.
    response_dict = xmltodict.parse(response.content)

    # This obscures StorageClass in a transformation, it is optional to add
    for item in response_dict['ListBucketResult']['Contents']:
        item['StorageClass'] = ""

    # Convert back to XML.
    listResultXml = xmltodict.unparse(response_dict)

    # Create response with listResultXml.
    response_with_list_result_xml = {
        'statusCode': 200,
        'listResultXml': listResultXml
    }

    # Create response with listBucketResult.
```

```
    response_dict['ListBucketResult'] =
 sanitize_response_dict(response_dict['ListBucketResult'])
    response_with_list_bucket_result = {
        'statusCode': 200,
        'listBucketResult': response_dict['ListBucketResult']
    }

    # Return the list to S3 Object Lambda.
    # Can return response_with_list_result_xml or response_with_list_bucket_result
    return response_with_list_result_xml

# Converting the response_dict's key to correct casing
def sanitize_response_dict(response_dict: dict):
    new_response_dict = dict()
    for key, value in response_dict.items():
        new_key = key[0].lower() + key[1:] if key != "ID" else 'id'
        if type(value) == list:
            newlist = []
            for element in value:
                if type(element) == type(dict()):
                    element = sanitize_response_dict(element)
                newlist.append(element)
            value = newlist
        elif type(value) == dict:
            value = sanitize_response_dict(value)
        new_response_dict[new_key] = value
    return new_response_dict
```

次の例は、ListObjectsV2 の Lambda レスポンス JSON の構造を示しています。

```
{
    "statusCode": <number>; // Required
    "errorCode": <string>;
    "errorMessage": <string>;
    "listResultXml": <string>; // This can also be Error XML string in case S3 returned
 error response when calling the pre-signed URL

    "listBucketResult": {  // listBucketResult can be provided instead of
 listResultXml, however they can not both be provided in the JSON response
        "name": <string>, // Required for 'listBucketResult'
        "prefix": <string>,
        "startAfter": <string>,
```

```
        "continuationToken": <string>,
        "nextContinuationToken": <string>,
        "keyCount": <int>, // Required for 'listBucketResult'
        "maxKeys": <int>, // Required for 'listBucketResult'
        "delimiter": <string>,
        "encodingType": <string>
        "isTruncated": <boolean>, // Required for 'listBucketResult'
        "contents": [ {
            "key": <string>, // Required for 'content'
            "lastModified": <string>,
            "eTag": <string>,
            "checksumAlgorithm": <string>, // CRC32, CRC32C, SHA1, SHA256
            "size": <int>, // Required for 'content'
            "owner": {
                "displayName": <string>, // Required for 'owner'
                "id": <string>, // Required for 'owner'
            },
            "storageClass": <string>
            },
            ...
        ],
        "commonPrefixes": [ {
            "prefix": <string> // Required for 'commonPrefix'
            },
        ...
        ],
    }
}
```

イベントコンテキストの形式と使用法

Amazon S3 Object Lambda は、AWS Lambda 関数に渡されたイベントで行われたリクエストに関するコンテキストを提供します。リクエストの例を次に示します。フィールドの説明は例の後に含まれています。

```
{
    "xAmzRequestId": "requestId",
    "getObjectContext": {
        "inputS3Url": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/example?X-Amz-Security-Token=<snip>",
        "outputRoute": "io-use1-001",
        "outputToken": "OutputToken"
    },
```

```
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
    "protocolVersion": "1.00"
}
```

リクエストには次のフィールドが含まれます。

- `xAmzRequestId` - このリクエストの Amazon S3 リクエスト ID。デバッグに役立つように、この値をログに記録することをお勧めします。
- `getObjectContext` - Amazon S3 および S3 Object Lambda への接続に関する入力および出力の詳細。
  - `inputS3Url` - Amazon S3 から元のオブジェクトを取得するために使用できる署名付き URL。URL は元の発信者のアイデンティティを使用して署名され、URL が使用されるときにそのユーザーのアクセス許可が適用されます。URL に署名されたヘッダーがある場合、Lambda 関数では、`Host` ヘッダーを除き、Amazon S3 への呼び出しにこれらのヘッダーを含める必要があります。
  - `outputRoute` – Lambda 関数が `WriteGetObjectResponse` を呼び出すときに S3 Object Lambda URL に追加されるルーティングトークン。
  - `outputToken` - `WriteGetObjectResponse` 呼び出しを元の発信者と一致させるために、S3 Object Lambda によって使用される不透明なトークン。
- `configuration` - Object Lambda アクセスポイントに関する設定情報。
  - `accessPointArn` - このリクエストを受信した Object Lambda アクセスポイントの Amazon リソースネーム (ARN)。
  - `supportingAccessPointArn` - Object Lambda アクセスポイント設定で指定されているサポートアクセスポイントの ARN。
  - `payload` - Object Lambda アクセスポイント設定に適用されるカスタムデータ。S3 Object Lambda はこのデータを不透明な文字列として扱うため、使用前にデコードする必要がある場合があります。
- `userRequest` - S3 Object Lambda への元の呼び出しに関する情報。
  - `url` - S3 Object Lambda で受信したリクエストのデコードされた URL。認可関連のクエリパラメータは除きます。
  - `headers` - HTTP ヘッダーと元の呼び出しからの値を含む文字列への文字列マップ。認可関連のヘッダーは含まれません。同じヘッダーが複数回表示される場合、同じヘッダーの各インスタンスからの値は、コンマ区切りのリストに結合されます。元のヘッダーの大文字と小文字は、このマップに保持されます。
- `userIdentity` – S3 Object Lambda への呼び出しを行ったアイデンティティの詳細。詳細については、AWS CloudTrail ユーザーガイドの「証跡へのデータイベントの口グ記録」を参照してください。
  - `type` – アイデンティティのタイプ。
  - `accountId` - アイデンティティが属するAWS アカウント。

  - `userName` - 呼び出しを行ったアイデンティティのわかりやすい名前。
  - `principalId` - 呼び出しを行ったアイデンティティの一意の識別子。
  - `arn` – 呼び出しを行ったプリンシパルの ARN。ARN の最後のセクションには、呼び出しを行ったユーザーまたはロールが含まれています。
  - `sessionContext` - リクエストが、一時的セキュリティ認証情報を使用して行われた場合、このエレメントはこれらの認証情報のために作成されたセッションに関する情報を提供します。
  - `invokedBy` – Amazon EC2 Auto Scaling や AWS Elastic Beanstalk など、リクエストを行った AWS のサービス の名前。
  - `sessionIssuer` - リクエストが一時的セキュリティ認証情報を使用して行われた場合、このエレメントは認証情報がどのように取得されたかに関する情報を提供します。
- `protocolVersion` – 提供されるコンテキストのバージョン ID。このフィールドの形式は `{Major Version}.{Minor Version}` です。マイナーバージョン番号は、常に 2 桁の数字です。フィールドのセマンティクスを削除または変更する場合は、メジャーバージョンバンプが必要となり、アクティブなオプトインが必要になります。Amazon S3 ではいつでも新しいフィールドを追加できます。この時点で、マイナーバージョンのバンプが発生する可能性があります。ソフトウェアのロールアウトの性質上、一度に複数のマイナーバージョンが使用されている場合があります。

Range および partNumber ヘッダーの操作

Amazon S3 Object Lambda で大規模なオブジェクトを操作する場合は、`Range` HTTP ヘッダーを使用して、オブジェクトから指定されたバイト範囲をダウンロードできます。同じオブジェクトのさまざまなバイト範囲をフェッチするには、Amazon S3 への同時接続を使用できます。また、オブジェクトの指定されたパートに対して範囲リクエストを実行する `partNumber` パラメータ(1 ~ 10,000 の整数) を指定することもできます。

`Range` または `partNumber` のパラメータを含んだリクエストを処理するには、複数の方法があるため、S3 オブジェクト Lambda ではこれらのパラメータを変換されたオブジェクトに適用しません。代わりに、AWS Lambda 関数は、アプリケーションで必要に応じてこの機能を実装する必要があります。

S3 Object Lambda で `Range` および `partNumber` パラメータを使用するには、次の操作を行います。

- Object Lambda アクセスポイントの設定でこれらのパラメータを有効にします。
- これらのパラメータを含むリクエストを処理できる Lambda 関数を作成します。

次のステップでそのやり方を説明します。

ステップ 1: Object Lambda アクセスポイントの設定

デフォルトでは、オブジェクト Lambda アクセスポイントは、ヘッダーまたはクエリパラメータに Range または partNumber パラメータを含む GetObject または HeadObject リクエストに対して、HTTP ステータスコード 501 (未実装) エラーで応答します。

Object Lambda アクセスポイントがこのようなリクエストを有効にするには、Object Lambda アクセスポイント設定の AllowedFeatures セクションに GetObject-Range、GetObject-PartNumber、HeadObject-Range、または HeadObject-PartNumber を含める必要があります。Object Lambda アクセスポイントの設定の更新の詳細については、「Object Lambda アクセスポイントの作成」を参照してください。

ステップ 2: Lambda 関数で **Range** または **partNumber** 処理を実装する

Object Lambda アクセスポイントが範囲 GetObject または HeadObject のリクエストで Lambda 関数を呼び出すとき、Range または partNumber パラメータはイベントコンテキストに含まれます。イベントコンテキストでのパラメータの場所は、次の表で説明するように、使用されたパラメータと Object Lambda アクセスポイントへの元のリクエストにどのように Lambda 含まれていたかによって異なります。

パラメータ	イベントコンテキストの場所
Range (ヘッダー)	userRequest.headers.Range
Range (クエリパラメータ)	userRequest.url (クエリパラメータ Range)
partNumber	userRequest.url (クエリパラメータ partNumber)

> **Important**
>
> 指定された Object Lambda アクセスポイントの署名付き URL には、元のリクエストの Range または partNumber パラメータが含まれていません。AWS Lambda 関数でこれらのパラメータを処理する方法については、以下のオプションを参照してください。

Range または partNumber の値を抽出した後に、アプリケーションのニーズに基づいて、次のいずれかの方法を使用できます。

A. リクエストされた **Range** または **partNumber** を変換されたオブジェクトにマッピングします (推奨)。

   Range または partNumber リクエストを処理する最も確実な方法は、以下を実行することです。

   - Amazon S3 から完全なオブジェクトを取得します。
   - オブジェクトを変換します。
   - リクエストされた Range または partNumber パラメータを変換後のオブジェクトに適用します。

   これを行うには、指定された署名付き URL を使用して Amazon S3 からオブジェクト全体をフェッチし、必要に応じてオブジェクトを処理します。例の Lambda 関数では、この方法で Range パラメータを設定し、AWS サンプル GitHub リポジトリの「このサンプル」を参照してください。

B. リクエストされた Range を署名付き URL にマッピングします。

   場合によっては、Lambda 関数でリクエストされた Range を署名済み URL に直接マッピングして、Amazon S3 からオブジェクトの一部のみを取得できます。このアプローチは、変換が次の両方の条件を満たしている場合にのみ適切です。

   1. 変換関数は、部分的なオブジェクト範囲に適用できます。
   2. 変換関数の前または後に Range パラメータを指定すると、同じ変換後のオブジェクトになります。

   たとえば、ASCII エンコードオブジェクト内のすべての文字を大文字に変換する変換関数は、上記の両方の条件を満たします。変換はオブジェクトの一部に適用でき、変換前に Range パラメータを適用すると、変換後にパラメータを適用するのと同じ結果が得られます。

   対照的に、ASCII エンコードされたオブジェクトの文字を反転する関数は、これらの条件を満たしていません。このような関数は、部分的なオブジェクト範囲に適用できるため、基準 1 を満たしています。ただし、基準 2 を満たしていません。なぜなら、Range パラメータを変換前に適用した場合と、変換後にパラメータを適用する場合とは結果が異なるためです。

   コンテンツ abcdefg を含むオブジェクトの最初の 3 文字に関数を適用するリクエストを考えてみましょう。変換前にの Range パラメータを適用すると abc のみが取得され、その後、データ

を逆にして戻すと cba が取得されます。しかし、変換後にパラメータが適用された場合、関数はオブジェクト全体を取得し、それを反転し、Range パラメータを適用して、gfe を返します。これらの結果は異なるため、この関数は Amazon S3 からオブジェクトを取得する際に、Range パラメータを適用すべきではありません。代わりに、オブジェクト全体を取得し、変換を実行してから、Range パラメータを適用する必要があります。

> **Warning**
>
> 多くの場合、Range パラメータを署名済み URL に適用すると、Lambda 関数またはリクエスト元のクライアントによる予期しない動作が発生します。Amazon S3 から部分的なオブジェクトのみを取得するときにアプリケーションが正常に動作することが確実でない限り、アプローチ A で前述したように、完全なオブジェクトを取得して変換することをお勧めします。

アプリケーションがアプローチ B の基準を満たしていれば、要求されたオブジェクト範囲のみをフェッチし、その範囲で変換を実行することで、AWS Lambda 関数を単純化することができます。

次の Java コードの例では、次の処理を実行する方法を示します。

- GetObject リクエストから Range ヘッダーを取得します。
- Lambda が Amazon S3 からリクエストされた範囲を取得するために使用できる署名付き URL に Range ヘッダーを追加します。

```
private HttpRequest.Builder applyRangeHeader(ObjectLambdaEvent event,
 HttpRequest.Builder presignedRequest) {
    var header = event.getUserRequest().getHeaders().entrySet().stream()
            .filter(e -> e.getKey().toLowerCase(Locale.ROOT).equals("range"))
            .findFirst();

    // Add check in the query string itself.
    header.ifPresent(entry -> presignedRequest.header(entry.getKey(),
 entry.getValue()));
    return presignedRequest;
}
```

AWS で構築された Lambda 関数の使用

AWS には、あらかじめ構築された AWS Lambda 関数が用意されています。これを Amazon S3 Object Lambda と共に使用して、個人を特定できる情報 (PII) を検出して編集したり、S3 オブジェクトを解凍したりできます。これらの Lambda 関数は AWS Serverless Application Repository にあります。これらの関数は、Object Lambda アクセスポイントの作成時に AWS Management Console から選択できます。

AWS Serverless Application Repository からサーバーレスアプリケーションをデプロイする方法の詳細については、「AWS Serverless Application Repository デベロッパーガイド」の「アプリケーションのデプロイ」を参照してください。

> **Note**
>
> 次の例は `GetObject` リクエストでのみ使用できます。

例 1: PII アクセスコントロール

Lambda 関数は Amazon Comprehend を使用します。これは、機械学習を使用してテキスト内でインサイトや関係性を検出する自然言語処理 (NLP) サービスです。この関数は、Amazon S3 バケット内のドキュメントから、名前、住所、日付、クレジットカード番号、社会保障番号などの個人を特定できる情報 (PII) を自動的に検出します。バケットに PII を含むドキュメントがある場合、PII アクセスコントロール関数を設定して、PII エンティティタイプを検出し、許可されていないユーザーへのアクセスを制限できます。

使用開始するには、アカウントに次の Lambda 関数をデプロイし、関数の Amazon リソースネーム (ARN) を Object Lambda アクセスポイント設定に追加します。

この関数の ARN の例を以下に示します。

```
arn:aws:serverlessrepo:us-east-1:111122223333:applications/
ComprehendPiiAccessControlS3ObjectLambda
```

AWS Management Console でこの関数をビューに追加するには、次の AWS Serverless Application Repository リンクを使用します。ComprehendPiiAccessControlS3ObjectLambda

GitHub でこの関数を表示するには、「Amazon Comprehend S3 Object Lambda」を参照してください。

例 2: PII Redaction

Lambda 関数は Amazon Comprehend を使用します。これは、機械学習を使用してテキスト内でインサイトや関係性を検出する自然言語処理 (NLP) サービスです。この関数は、Amazon S3 バケット内のドキュメントから、名前、住所、日付、クレジットカード番号、社会保障番号などの個人を特定できる情報 (PII) を自動的にマスキングします。

クレジットカード番号や銀行口座情報などの情報を含むドキュメントがバケットにある場合は、PII Redaction S3 Object Lambda 関数を設定して、PII を検出し、PII エンティティタイプがマスキングされたドキュメントのコピーを返すように設定できます。

使用開始するには、アカウントに次の Lambda 関数をデプロイし、関数の ARN を Object Lambda アクセスポイント設定に追加します。

この関数の ARN の例を以下に示します。

```
arn:aws:serverlessrepo:us-east-1:111122223333::applications/
ComprehendPiiRedactionS3ObjectLambda
```

AWS Management Console にこの関数を表示または追加するには、次の AWS Serverless Application Repository リンクを使用します。ComprehendPiiRedactionS3ObjectLambda

GitHub でこの関数を表示するには、「Amazon Comprehend S3 Object Lambda」を参照してください。

PII リダクションにおけるいくつかの S3 Object Lambda タスクにおけるエンドツーエンドの一連の手順については、「チュートリアル: S3 Object Lambda と Amazon Comprehend を使用した PII データの検出と編集」を参照してください。

例 3: 解凍

Lambda 関数 `S3ObjectLambdaDecompression` は、Amazon S3 に格納されているオブジェクトを、次の 6 つの圧縮ファイル形式のうちの 1 つで解凍できます。`bzip2`、`gzip`、`snappy`、`zlib`、`zstandard`、および `ZIP`。

使用開始するには、アカウントに次の Lambda 関数をデプロイし、関数の ARN を Object Lambda アクセスポイント設定に追加します。

この関数の ARN の例を以下に示します。

```
arn:aws:serverlessrepo:us-east-1:111122223333::applications/S3ObjectLambdaDecompression
```

AWS Management Console でこの関数をビューに追加するには、次の AWS Serverless Application Repository リンクを使用します。S3ObjectLambdaDecompression

GitHub でこの関数を表示するには、「S3 Object Lambda Decompression」(S3 Object Lambda の解凍) を参照してください。

S3 Object Lambda のベストプラクティスとガイドライン

S3 Object Lambda を使用する場合は、操作とパフォーマンスを最適化するには、以下のベストプラクティスとガイドラインに従ってください。

トピック

S3 Object Lambda の使用

S3 Object Lambda は、GET、LIST、および HEAD リクエストの処理のみをサポートします。他のリクエストは AWS Lambda を呼び出さず、標準の変換されていない API レスポンスを返します。リージョンごとに AWS アカウント あたり最大 1,000 個の Object Lambda アクセスポイントを作成できます。使用する AWS Lambda 関数は、Object Lambda アクセスポイントと同じ AWS アカウントおよびリージョンに存在する必要があります。

S3 Object Lambda では、発信者に完全なレスポンスをストリームするのに最大 60 秒かかります。関数には、AWS Lambda のデフォルトクォータも適用されます。詳細については、AWS Lambda デベロッパーガイドの Lambda のクォータを参照してください。

S3 Object Lambda が指定された Lambda 関数を呼び出す場合、データが指定された Lambda 関数またはアプリケーションによって意図的に上書きまたは Amazon S3 から削除され、それが正しいことを確認する責任は、ユーザーにあります。

S3 Object Lambda は、オブジェクトに対する操作の実行にのみ使用できます。S3 Object Lambda を使用して、バケットの変更や削除など、他の Amazon S3 オペレーションを実行することはできません。アクセスポイントをサポートする S3 オペレーションの詳細なリストについては、「S3 オペレーションとアクセスポイントの互換性」を参照してください。

このリストに加えて、Object Lambda アクセスポイントは POST Object、CopyObject (ソースとして)、および SelectObjectContent API オペレーションをサポートしていません。

S3 Object Lambda に関連して使用される AWS のサービス

S3 Object Lambda は、Amazon S3、AWS Lambda、およびオプションで、リクエストしているアプリケーションに関連するオブジェクトを配信するために選択した他の AWS のサービス を接続します。S3 Object Lambda で使用されるすべての AWS のサービス は、それぞれのサービスレベルアグリーメント (SLA) に準拠します。例えば、AWS のサービス がサービスコミットメントを満たさない場合は、そのサービスの SLA に記されたとおり、ユーザーにはサービスクレジットを受け取る資格が発生します。

`Range` および `partNumber` ヘッダー

大規模なオブジェクトを操作する場合は、Range HTTP ヘッダーを使用して、オブジェクトから指定されたバイト範囲をダウンロードできます。Range ヘッダーを使用すると、リクエストはオブジェクトの指定された部分のみをフェッチします。partNumber ヘッダーは、オブジェクトから指定されたパートに対して範囲リクエストを実行するためにも使用できます。

詳細については、「Range および partNumber ヘッダーの操作」を参照してください。

`expiry-date` の変換

AWS Management Console の Object Lambda アクセスポイントから、変換されたオブジェクトを開いたり、ダウンロードしたりできます。これらのオブジェクトは、期限が切れていないものである必要があります。Lambda 関数でオブジェクトの expiry-date を変換すると、開いたりダウンロードしたりできない期限切れのオブジェクトが表示されることがあります。この動作は、S3 Glacier Flexible Retrieval、および S3 Glacier Deep Archive の復元されたオブジェクトにのみ適用されます。

AWS CLI と AWS SDK の使用

AWS Command Line Interface (AWS CLI) S3 サブコマンド (cp、mv、および sync)、および AWS SDK for Java TransferManager クラスについては、S3 Object Lambda での使用はサポートされていません。

S3 Object Lambda のチュートリアル

次のチュートリアルでは、いくつかの S3 Object Lambda タスクにおけるエンドツーエンドの一連の手順について説明します。

- チュートリアル: S3 Object Lambda を使用したアプリケーションのデータの変換
- チュートリアル: S3 Object Lambda と Amazon Comprehend を使用した PII データの検出と編集
- チュートリアル: S3 Object Lambda を使用して、取得時に画像に動的に透かしを入れる

S3 Object Lambda のデバッグ

Amazon S3 Object Lambda アクセスポイントへのリクエストにより、Lambda 関数の呼び出しまたは実行で何か問題が発生したときに、新しいエラーレスポンスが発生することがあります。このようなエラーは、標準の Amazon S3 エラーと同じ形式に従います。S3 Object Lambda エラーについては、Amazon Simple Storage Service API リファレンスの S3 Object Lambda エラーコードリストを参照してください。

一般的な Lambda 関数のデバッグの詳細については、AWS Lambda デベロッパーガイドの「Lambda アプリケーションのモニタリングとトラブルシューティング」を参照してください。

Amazon S3 の標準エラーについては、Amazon Simple Storage Service API リファレンスのエラーレスポンスを参照してください。

Object Lambda アクセスポイントに対して Amazon CloudWatch のリクエストメトリクスを有効にできます。このメトリクスは、アクセスポイントの動作パフォーマンスのモニタリングに役立ちます。リクエストメトリクスは、Object Lambda アクセスポイントの作成中もしくは作成後に有効にできます。詳細については、「CloudWatch の S3 Object Lambda リクエスト」を参照してください。

Object Lambda アクセスポイントに対して行われたリクエストについて、より詳細なログを取得するには、AWS CloudTrail データイベントを有効にできます。詳細については、AWS CloudTrail ユーザーガイドの「証跡へのデータイベントのログ記録」を参照してください。

S3 Object Lambda チュートリアルについては、以下を参照してください。

- チュートリアル: S3 Object Lambda を使用したアプリケーションのデータの変換
- チュートリアル: S3 Object Lambda と Amazon Comprehend を使用した PII データの検出と編集
- チュートリアル: S3 Object Lambda を使用して、取得時に画像に動的に透かしを入れる

標準のアクセスポイントの詳細については、Amazon S3 アクセスポイントを使用したデータアクセスの管理 を参照してください。

バケットの操作方法の詳細については、「バケットの概要」を参照してください。オブジェクトの操作方法の詳細については、「Amazon S3 オブジェクトの概要」を参照してください。

S3 Express One Zone とは

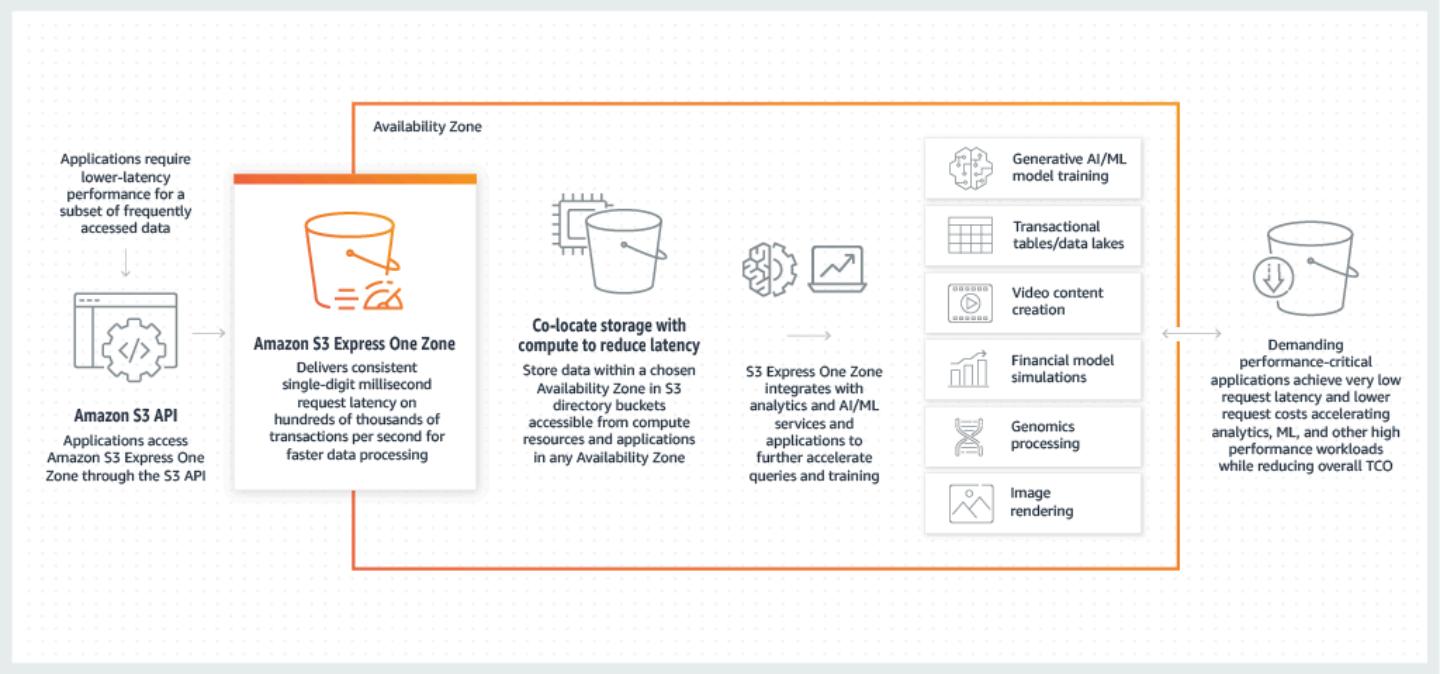
Amazon S3 Express One Zone は、最もレイテンシーの影響を受けやすいアプリケーションに 1 桁のミリ秒単位で一貫したデータアクセスを提供することを目的として構築された、高パフォーマンスのシングルアベイラビリティーゾーンの Amazon S3 ストレージ クラスです。S3 Express One Zone は、現在入手可能なレイテンシーが最も低いクラウドオブジェクトストレージクラスで、データアクセス速度は最大 10 倍速く、リクエストコストは S3 Standard より 50% 低減されます。リクエストが最大で 1 桁速く完了するため、アプリケーションは直ちにメリットを得られます。S3 Express One Zone は、その他の S3 ストレージクラスと同様のパフォーマンス伸縮性を備えています。

他の Amazon S3 ストレージクラスと同様に、容量やスループットの要件を事前に計画したりプロビジョンしたりする必要はありません。必要に応じてストレージをスケールアップまたはスケールダウンでき、Amazon S3 API を介してデータにアクセスできます。

S3 Express One Zone は、オブジェクトストレージをコンピューティングリソースと同じ場所に配置するオプションを提供し、単一のアベイラビリティーゾーンを選択できる最初の S3 ストレージクラスです。これにより、アクセス速度が最大限に高速化します。また、アクセス速度をさらに向上させて 1 秒あたり数十万のリクエストに対応するために、S3 Express One Zone ストレージクラスはデータを新しいバケットタイプである Amazon S3 ディレクトリバケットに保存します。キー名やアクセスパターンを問わず、各ディレクトリバケットは、1 秒あたり数十万のトランザクション (TPS) をサポートできます。

Amazon S3 Express One Zone ストレージクラスは、単一のアベイラビリティーゾーン内で 99.95% の可用性を実現するように設計されており、Amazon S3 サービスレベル契約に基づいています。S3 Express One Zone では、データは単一のアベイラビリティーゾーン内で冗長的に複数のデバイスに保存されます。S3 Express One Zone は、冗長性の損失を迅速に検出して修復することによって、同時デバイス障害を処理するように設計されています。既存のデバイスに障害が発生した場合、S3 Express One Zone はアベイラビリティーゾーン内の新しいデバイスに自動的にリクエストを移します。この冗長性により、アベイラビリティーゾーン内のデータへのアクセスの中断が回避されます。

S3 Express One Zone は、オブジェクトへのアクセスに必要なレイテンシーを最小限に抑えることが重要なあらゆるアプリケーションに最適です。このようなアプリケーションは、クリエイティブプロフェッショナルがユーザーインターフェイスからコンテンツに迅速にアクセスする必要がある、動画編集のような人間との対話型のワークフローになる可能性があります。S3 Express One Zone は、データに対する応答性の要件が似ている分析や機械学習のワークロード、特に小規模なアクセスやランダムアクセスが多いワークロードにも役立ちます。S3 Express One Zone は、Amazon EMR、Amazon SageMaker、Amazon Athena など、分析や人工知能と機械学習 (AI/ML) のワークロードをサポートするためにその他の AWS のサービス と併用できます。



S3 Express One Zone を使用する場合、ゲートウェイ VPC エンドポイントを使用して、仮想プライベートクラウド (VPC) 内のディレクトリバケットを操作できます。ゲートウェイエンドポイントを使用すると、VPC 用のインターネットゲートウェイや NAT デバイスを必要とせず追加コストあんしで、VPC から S3 Express One Zone にディレクトリバケットにアクセスできます。

汎用バケットやその他のストレージクラスで使用しているのと同じ Amazon S3 の API オペレーションと機能の多くをディレクトリバケットでも使用できます。これには、Amazon S3 のマウントポイント、Amazon S3 マネージドキー (SSE-S3) によるサーバー側の暗号化、S3 バッチオペレーション、S3 ブロックパブリックアクセスなどがあります。Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、AWS SDK、Amazon S3 REST API を使用して S3 Express One Zone にアクセスすることができます。

S3 Express One Zone の詳細については、次のトピックを参照してください。

- 概要
- S3 Express One Zone の機能
- 関連サービス
- 次のステップ

概要

パフォーマンスを最適化してレイテンシーを低減するために、S3 Express One Zone には次の新しい概念が導入されています。

単一のアベイラビリティーゾーン

Amazon S3 Express One Zone ストレージクラスは、単一のアベイラビリティーゾーン内で 99.95% の可用性を実現するように設計されており、Amazon S3 サービスレベル契約に基づいています。S3 Express One Zone では、データは単一のアベイラビリティーゾーン内で冗長的に複数のデバイスに保存されます。S3 Express One Zone は、冗長性の損失を迅速に検出して修復することによって、同時デバイス障害を処理するように設計されています。既存のデバイスに障害が発生した場合、S3 Express One Zone はアベイラビリティーゾーン内の新しいデバイスに自動的にリクエストを移します。この冗長性により、アベイラビリティーゾーン内のデータへのアクセスの中断が回避されます。

アベイラビリティーゾーンは、AWS リージョン の 冗長電源、ネットワーク、および接続を備えた1つ以上の個別のデータセンターです。ディレクトリバケットを作成する際は、バケットの配置場所であるアベイラビリティーゾーンと AWS リージョン を選択します。

ディレクトリバケット

Amazon S3 バケットには、S3 汎用バケットと S3 ディレクトリバケットの 2 種類があります。汎用バケットは、S3 ユースケースの大部分で使用されているデフォルトの Amazon S3 バケットタイプです。ディレクトリバケットは、S3 Express One Zone ストレージクラスのみを使用します。これは、一貫して 1 桁ミリ秒のレイテンシーに維持する必要があるワークロードまたはパフォーマンス重視のアプリケーション向けに設計されています。アプリケーションとパフォーマンス要件に最適なバケットタイプを選択します。

ディレクトリバケットは、汎用バケットのフラットなストレージ構造とは対照的に、データを階層的にディレクトリにまとめます。ディレクトリバケットにはプレフィックスの制限はなく、個々のディレクトリは水平方向にスケールできます。

ディレクトリバケットは、パフォーマンス重視のアプリケーションで使用されるように構築された S3 Express One Zone ストレージクラスを使用します。S3 Express One Zone は、オブジェクトストレージをコンピュートリソースと同じ場所に配置するオプションを備えた単一のアベイラビリティーゾーンを選択できるため、最高レベルのアクセス速度を実現できます。これは、AWS リージョン 内の複数のアベイラビリティーゾーンでオブジェクトを冗長的に保存する汎用バケットとは異なります。

汎用バケットの詳細については、「ディレクトリバケット」を参照してください。汎用バケットの詳細については、「バケットの概要」を参照してください。

エンドポイントとゲートウェイ VPC エンドポイント

ディレクトリバケットのバケット管理 API オペレーションはリージョンエンドポイントを通じて利用でき、リージョンエンドポイント API オペレーションと呼ばれます。リージョンエンドポイント API オペレーションの例には、`CreateBucket`、`DeleteBucket` があります。ディレクトリバケットを作成した後、ゾーンエンドポイント API オペレーションを使用して、ディレクトリバケット内のオブジェクトをアップロードして管理できます。ゾーンエンドポイント API オペレーションは、ゾーンエンドポイントを通じて実行できます。ゾーン別エンドポイント API オペレーションの例には、`PutObject`、`CopyObject` があります。

ゲートウェイ VPC エンドポイントを使用して、VPC から S3 Express One Zone にアクセスできます。ゲートウェイ エンドポイントを作成した後、VPC から S3 Express One Zone を送信先とするトラフィックのルートテーブルにターゲットとして追加できます。Amazon S3 と同様、ゲートウェイエンドポイントの使用に追加料金は発生しません。ゲートウェイ VPC エンドポイントの設定方法の詳細については、「S3 Express One Zone のネットワーク」を参照してください。

セッションベースの承認

S3 Express One Zone では、レイテンシーを最小限に抑えるように最適化された新しいセッションベースのメカニズムを使用して、リクエストを認証および承認します。`CreateSession` を使用して、バケットへの低レイテンシーアクセスを実現する一時的な認証情報をリクエストできます。このような一時的な認証情報は、特定の S3 ディレクトリバケットに限定されます。セッショントークンはゾーン (オブジェクトレベル) オペレーション (CopyObject を除く) でのみ使用されます。詳細については、「`CreateSession` authorization」を参照してください。

S3 Express One Zone でサポートされている AWS SDK は、ユーザーに代わってセッションの確立と更新を処理します。セッションを保護するため、一時的なセキュリティ認証情報は 5 分後に期限切れになります。AWS SDK をダウンロードしてインストールし、必要な AWS Identity and Access Management (IAM) アクセス許可を設定したら、直ちに API オペレーションの使用を開始できます。

S3 Express One Zone の機能

S3 Express One Zone では、次の S3 の機能が利用できます。サポートされている API とサポートされていない API の全リストについては、「S3 Express One Zone の違いとは」を参照してください。

アクセス管理とセキュリティ

ディレクトリバケットでは、次の機能を使用してアクセスを監査および管理することができます。デフォルトでは、ディレクトリバケットはプライベートであり、アクセスを明示的に許可されているユーザーのみがアクセスできます。バケット、プレフィックス、またはオブジェクトタグレベルでアクセス制御境界を設定できる汎用バケットとは異なり、ディレクトリバケットのアクセス制御境界はバケットレベルでのみ設定されます。詳細については、「S3 Express One Zone 向け AWS Identity and Access Management (IAM)」を参照してください。

- S3 ブロックパブリックアクセス – すべての S3 ブロックパブリックアクセス設定は、デフォルトでバケットレベルで有効になっています。このデフォルトの設定は変更できません。
- S3 オブジェクト所有権 (デフォルトでバケット所有者の強制) – アクセスコントロールリスト (ACL) はディレクトリバケットではサポートされていません。ディレクトリバケットは、S3 オブジェクト所有権のバケット所有者による強制設定を自動的に使用します。バケット所有者の強制とは、ACL が無効になり、バケット所有者はバケット内のすべてのオブジェクトを自動的に所有し、完全にコントロールできることを意味します。このデフォルトの設定は変更できません。
- AWS Identity and Access Management (IAM) – IAM は、ディレクトリバケットへのアクセスを安全に制御するうえで役立ちます。IAM を使用すると、`s3express:CreateSession` アクションを通じてバケット管理 (リージョン) API オペレーションとオブジェクト管理 (ゾーン) API オペレーションへのアクセスを許可できます。詳細については、「S3 Express One Zone 向け AWS Identity and Access Management (IAM)」を参照してください。オブジェクト管理アクションとは異なり、バケット管理アクションはクロスアカウント間で実行できません。バケット所有者のみが、このようなアクションを実行できます。
- バケットポリシー – IAM ベースのポリシー言語を使用して、ディレクトリバケットのリソースベースのアクセス許可を設定します。IAM を使用して `CreateSession` API オペレーションへのアクセスを制御することもできます。これにより、ゾーン API オペレーションまたはオブジェクト管理 API オペレーションを使用できるようになります。ゾーン API オペレーションに同一アカウントまたはクロスアカウントのアクセス許可を付与することができます。S3 Express One Zone のアクセス許可とポリシーの詳細については、「S3 Express One Zone 向け AWS Identity and Access Management (IAM)」を参照してください。
- IAM Access Analyzer for S3 – アクセスポリシーを評価してモニタリングし、ポリシーが S3 リソースへの意図したアクセスのみを提供していることを確認します。

ロギングとモニタリング

S3 Express One Zone は、リソースの使用状況のモニタリングと制御に利用できる次のような S3 ログと監視ツールを使用します。

- Amazon CloudWatch metrics – CloudWatch を使用して AWS リソースとアプリケーションをモニタリングして、パフォーマンスのメトリクスを収集し、追跡します。S3 Express One Zone は、その他の Amazon S3 ストレージクラス (`AWS/S3`) と同じ CloudWatch 名前空間を使用し、`BucketSizeBytes` と `NumberOfObjects` のディレクトリバケットの毎日のストレージメトリクスをサポートします。詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。
- AWS CloudTrail ログ – AWS CloudTrail は、ユーザー、ロール、または AWS のサービス が実行するアクションを記録して、AWS アカウント のオペレーションとリスクの監査、ガバナンス、コンプライアンスの実施に役立つ AWS のサービス です。S3 Express One Zone の場合、CloudTrail はリージョンのエンドポイント API オペレーション (`CreateBucket` や `PutBucketPolicy`など) を管理イベントとしてキャプチャします。これらのイベントには、AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、および AWS API オペレーションで実行されたアクションが含まれます。S3 Express One Zone の CloudTrail 管理イベントの `eventsource` は、`s3express.amazonaws.com` です。詳細については、「Amazon S3 CloudTrail イベント」を参照してください。

> **Note**
>
> Amazon S3 サーバーのアクセスログは S3 Express One Zone ではサポートされていません。

オブジェクト管理

ディレクトリバケットを作成すると、Amazon S3 コンソール、AWS SDK、AWS CLI を使用してオブジェクトストレージを管理できるようになります。S3 Express One Zone でのオブジェクト管理には次の機能を使用できます。

- S3 バッチオペレーション – バッチオペレーションを使用して、ディレクトリバケット内のオブジェクトに対して一括オペレーションを実行します。例には、Copy、Invoke AWS Lambda 関数などがあります。例えばバッチオペレーションを使用して、ディレクトリバケットと汎用バケット間でオブジェクトをコピーできます。バッチオペレーションを使用すると、AWS SDK または

AWS CLI を使用するか、Amazon S3 コンソールで数回クリックするだけで、単一の S3 リクエストで数十億のオブジェクトを大規模に管理できます。

- インポート – ディレクトリバケット作成後、Amazon S3 コンソールのインポート機能を使用してバケットにオブジェクトを追加できます。インポートは、汎用バケットからディレクトリバケットにオブジェクトをコピーするバッチオペレーションジョブを作成するための効率的な方法です。

AWS SDK とクライアントライブラリ

ディレクトリバケットを作成して、バケットにオブジェクトをアップロードした後、次を使用してオブジェクトストレージを管理できます。

- Mountpoint for Amazon S3 – Mountpoint for Amazon S3 は、高スループットアクセスを実現し、Amazon S3 データレイクのコンピューティングコストを削減するオープンソースのファイルクライアントです。Mountpoint for Amazon S3 は、ローカルファイルシステムの API コールを `GET` や `LIST` などの S3 オブジェクト API コールに変換します。ペタバイトのデータを処理できるため、Amazon S3 が提供する伸縮性に優れたスループットを必要とする、読み取り量の多いデータレイクワークロードに最適です。これにより、数千のインスタンスにわたってスケールアップしたりスケールダウンしたりできます。
- S3A – S3Aは Amazon S3 のデータストアにアクセスするための Hadoop と互換性のある推奨インターフェイスです。S3A は Hadoop の S3N ファイルシステムクライアントの後継です。
- PyTorch on AWS — PyTorch on AWS は、機械学習モデルの開発と本番環境へのデプロイを容易にするオープンソースのディープラーニングフレームワークです。
- AWS SDK – Amazon S3 でのアプリケーション開発には AWS SDK を使用できます。AWS SDK は、基盤となる Amazon S3 REST API をラップして、プログラミング作業を簡素化します。S3 Express One Zone での AWS SDK の使用の詳細については、「the section called “AWS SDK”」を参照してください。

暗号化とデータ保護

ディレクトリバケットに保存されているオブジェクトは、Amazon S3 マネージドキー (SSE-S3) を使用したサーバー側の暗号化を使用して自動的に暗号化されます。ディレクトリバケットは、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、お客様が提供する暗号化キーによるサーバー側の暗号化 (SSE-C)、または AWS KMS keys キーによる二層式サーバー側の暗号化 (DSSE-KMS) をサポートしていません。詳細については、データ保護と暗号化およびAmazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)を参照してください。

S3 Express One Zone では、アップロードまたはダウンロード中にデータを検証するために使用されるチェックサムアルゴリズムを選択するオプションが提供されます。CRC32、CRC32C、SHA-1、SHA-256 などのセキュアハッシュアルゴリズム (SHA) や巡回冗長検査 (CRC) データ整合性チェックアルゴリズムのいずれかを選択できます。MD5 ベースのチェックサムは S3 Express One Zone ストレージクラスではサポートされていません。

詳細については、「S3 の追加のチェックサムのベストプラクティス」を参照してください。

AWS Signature Version 4 (SigV4)

S3 Express One Zone は AWS Signature Version 4 (SigV4) を使用します。SigV4 は Amazon S3 に対する HTTP 経由のリクエストを認証するために使用される署名プロトコルです。S3 Express One Zone は AWS Sigv4 を使用してリクエストに署名します。詳細については、Amazon Simple Storage Service API リファレンスの「リクエストの認証 (AWS 署名バージョン 4)」を参照してください。

強力な整合性

S3 Express One Zone は、すべての AWS リージョン のディレクトリバケット内のオブジェクトの PUT と DELETE リクエストに対して、読み取り後書き込みの強力な一貫性を提供します。詳細については、「Amazon S3 のデータ整合性モデル」を参照してください。

関連サービス

S3 Express One Zone AWS のサービス ストレージクラスで次を使用して、特定の低レイテンシーのユースケースをサポートできます。

- Amazon Elastic Compute Cloud (Amazon EC2) – Amazon EC2 は、安全でスケーラブルなコンピューティング容量を AWS クラウド で提供します。Amazon EC2 の使用により、ハードウェアに事前投資する必要がなくなり、アプリケーションをより速く開発およびデプロイできます。Amazon EC2 を使用すると、必要な数 (またはそれ以下) の仮想サーバーの起動、セキュリティおよびネットワーキングの構成、ストレージの管理ができます。
- AWS Lambda – Lambda はサーバーのプロビジョニングや管理を必要とせずにコードを実行できるコンピューティングサービスです。バケットの通知設定を構成し、関数のリソースベースのアクセス許可ポリシーで関数を呼び出すためのアクセス許可を Amazon S3 に付与します。
- Amazon Elastic Kubernetes Service (Amazon EKS) は、AWS 上で、独自の Kubernetes コントロールプレーンをインストール、運用、保守する必要がないマネージド型サービスです。Kubernetes は、コンテナ化されたアプリケーションの管理、スケーリング、デプロイを自動化するオープンソースシステムです。

- Amazon Elastic Container Service (Amazon ECS) – Amazon ECS は、コンテナ化されたアプリケーションを簡単にデプロイ、管理、スケーリングできる、フルマネージドコンテナオーケストレーションサービスです。
- Amazon Athena – Athena はインタラクティブなクエリサービスで、Amazon S3 内のデータを標準 SQL を使用して直接的に簡単に分析できます。また、Athena を使用すると、リソースの計画、設定、管理を必要とせずに Apache Spark を使用してデータ分析をインタラクティブに実行することもできます。Athena で Apache Spark アプリケーションを実行する場合は、処理用の Spark コードを送信して、結果を直接受け取ります。
- Amazon SageMaker Runtime Model Training – Amazon SageMaker Runtime は、フルマネージド型の機械学習サービスです。SageMaker Runtime では、データサイエンティストやデベロッパーが迅速かつ簡単に機械学習モデルの構築とトレーニングを行うことができ、それらを稼働準備が整ったホストされている環境に直接デプロイできます。
- AWS Glue – AWS Glue は、分析を行うユーザーが複数のソースからのデータを簡単に検出、準備、移動、統合できるようにするサーバーレスのデータ統合サービスです。分析用の AWS Glue は、分析、機械学習、アプリケーション開発に使用できます。AWS Glue には、ジョブの作成、実行、ビジネスワークフローの実装のための生産性向上ツールとデータ運用ツールも追加されています。
- Amazon EMR – Amazon EMR は、Apache Hadoop や Apache Spark などのビッグデータフレームワークを AWS で簡単に実行して、膨大な量のデータを処理および分析できるマネージドクラスタープラットフォームです。

次のステップ

S3 Express One Zone ストレージクラスとディレクトリバケットの使用の詳細については、次のトピックを参照してください。

- S3 Express One Zone の違いとは
- S3 Express One Zone の使用を開始する
- S3 Express One Zone のネットワーク
- ディレクトリバケット
- ディレクトリバケットでのオブジェクトの使用
- S3 Express One Zone のセキュリティ
- Amazon S3 Express One Zone のパフォーマンスの最適化
- S3 Express One Zone を使用した開発

S3 Express One Zone の違いとは

Amazon S3 Express One Zone は、最もレイテンシーの影響を受けやすいアプリケーションに 1 桁のミリ秒単位で一貫したデータアクセスを提供することを目的として構築された、高パフォーマンスのシングルアベイラビリティーゾーンの Amazon S3 ストレージ クラスです。S3 Express One Zone は、オブジェクトストレージをコンピュートリソースと同じ場所に配置するオプションを備えた単一のアベイラビリティーゾーンを選択できる最初の S3 ストレージクラスです。これにより、最高レベルのアクセス速度を実現できます。また、アクセス速度をさらに向上し、1 秒あたり数十万のリクエストをサポートするために、S3 Express One Zone はデータを新しいバケットタイプである Amazon S3 ディレクトリバケットに保存します。

詳細については、S3 Express One Zone とおよびディレクトリバケットを参照してください。

Amazon S3 API を使用して、ディレクトリバケットを作成し、S3 Express One Zone のデータにアクセスできます。Amazon S3 API は S3 Express One Zone やディレクトリバケットと互換性がありますが、いくつか大きな違いがあります。S3 Express One Zone の違いの詳細については、次のトピックを参照してください。

トピック

- S3 Express One Zone の違い
- S3 Express One Zone がサポートする API オペレーション
- S3 Express One Zone がサポートしていない Amazon S3 の機能

S3 Express One Zone の違い

- サポートされているバケットタイプ – S3 Express One Zone ストレージクラスのオブジェクトは、ディレクトリバケットにのみ保存できます。詳細については、「ディレクトリバケット」を参照してください。
- 永続性 – S3 Express One Zone では、データは単一のアベイラビリティーゾーン内で冗長的に複数のデバイスに保存されます。S3 Express One Zone は、単一のアベイラビリティーゾーン内で 99.95% の可用性を実現するように設計されており、Amazon S3 サービスレベル契約に基づいています。詳細については、「単一のアベイラビリティーゾーン」を参照してください。
- **`ListObjectsV2`** の動作
  - ディレクトリバケットの場合、`ListObjectsV2` は辞書順 (アルファベット順) にオブジェクトを返すわけではありません。さらに、プレフィックスは区切り文字で終わる必要があり、区切り文字として指定できるのは「/」のみです。

  - ディレクトリバケットの場合、ListObjectsV2 レスポンスには、進行中のマルチパートアップロードにのみ関連するプレフィックスが含まれます。
- 削除動作 – ディレクトリバケット内のオブジェクトを削除すると、Amazon S3 はオブジェクトパス内の空のディレクトリをすべて再帰的に削除します。例えば、オブジェクトキー dir1/dir2/file1.txt を削除した場合、Amazon S3 は file1.txt を削除します。 dir1/ と dir2/ のディレクトリが空で、その他のオブジェクトが含まれていない場合、Amazon S3 はこれらのディレクトリも削除します。
- ETag とチェックサム – S3 Express One Zone のエンティティタグ (ETag) はランダムな英数字文字列であり、MD5 チェックサムではありません。S3 Express One Zone での追加のチェックサムの詳細については、「S3 の追加のチェックサムのベストプラクティス」を参照してください。
- **DeleteObjects** リクエスト内のオブジェクトキー
  - DeleteObjects リクエストのオブジェクトキーには、空白以外の文字を 1 つ以上含める必要があります。すべての空白文字を含む文字列は DeleteObjects リクエストではサポートされていません。
  - DeleteObjects リクエストのオブジェクトキーには、改行 (\n)、タブ (\t) とキャリッジリターン (\r) 文字を除き、Unicode 制御文字を含めることはできません。
- リージョンエンドポイントとゾーンエンドポイント – S3 Express One Zone を使用する場合、すべてのクライアントリクエストでリージョンを指定する必要があります。リージョンエンドポイントの場合は、s3express-control.us-west-2.amazonaws.com などのリージョンを指定します。ゾーンエンドポイントの場合は、s3express-usw2-az1.us-west-2.amazonaws.com など、リージョンとアベイラビリティーゾーンの両方を指定します。詳細については、「リージョンエンドポイントとゾーンエンドポイント」を参照してください。
- マルチパートアップロード – Amazon S3 に保存されているその他のオブジェクトと同様に、マルチパートアップロードプロセスを使用して S3 Express One Zone ストレージクラスに保存されているサイズの大きいオブジェクトをアップロードしてコピーできます。ただし、S3 Express One Zone に保存されているオブジェクトでマルチパートアップロードプロセスを使用する場合は次のような違いがあります。詳細については、「the section called “ディレクトリバケットでのマルチパートアップロードの使用”」を参照してください。
  - オブジェクトの作成日はマルチパートアップロードの完了日となります。
  - マルチパート番号には連続したパート番号を使用する必要があります。連続しないパーツ番号を使用してマルチパートアップロードリクエストを実行しようとすると、Amazon S3 は HTTP 400 (Bad Request) エラーを生成します。
  - マルチパートアップロードを開始するユーザーは、s3express:CreateSession アクセス許可を通じて明示的に AbortMultipartUpload へのアクセス許可が付与されている場合にの

み、マルチパートアップロードリクエストを中止できます。詳細については、「S3 Express One Zone 向け AWS Identity and Access Management (IAM)」を参照してください。

- ディレクトリバケットを空にする – AWS Command Line Interface (CLI) による `s3 rm` コマンド、Mountpoint による `delete` オペレーション、AWS Management Console による空のバケットオプションボタンでは、ディレクトリバケット内の進行中のマルチパートアップロードを削除できません。こうした進行中のマルチパートアップロードを削除するには、`ListMultipartUploads` オペレーションを使用してバケット内の進行中のマルチパートアップロードを一覧表示し、`AbortMultupartUpload` オペレーションを使用して進行中のすべてのマルチパートアップロードを中止します。

S3 Express One Zone がサポートする API オペレーション

Amazon S3 Express One Zone ストレージクラスは、リージョン (バケットレベル、またはコントロールプレーン) とゾーン (オブジェクトレベル、またはデータ プレーン) の両方のエンドポイント API オペレーションをサポートします。詳細については、S3 Express One Zone のネットワークおよびエンドポイントとゲートウェイ VPC エンドポイントを参照してください。

リージョンエンドポイント API オペレーション

S3 Express One Zone では、次のリージョンエンドポイント API オペレーションがサポートされています。

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets
- PutBucketPolicy

ゾーンエンドポイント API オペレーション

S3 Express One Zone では、次のゾーンエンドポイント API オペレーションがサポートされています。

- CreateSession
- CopyObject

- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- PutObject
- AbortMultipartUpload
- CompleteMultiPartUpload
- CreateMultipartUpload
- ListMultipartUploads
- ListParts
- UploadPart
- UploadPartCopy

S3 Express One Zone がサポートしていない Amazon S3 の機能

S3 Express One Zone では、次の Amazon S3 の機能がサポートされていません。

- AWS CloudTrail データプレーンイベント
- AWS マネージドポリシー
- S3 向け AWS PrivateLink
- MD5 チェックサム
- 多要素認証 (MFA) Delete
- S3 オブジェクトロック
- リクエスタ支払い
- S3 Access Grants
- S3 アクセスポイント
- バケットのタグ

- Amazon CloudWatch のリクエストメトリクス
- S3 イベント通知
- S3 ライフサイクル
- S3 マルチリージョンアクセスポイント
- S3 Object Lambda アクセスポイント
- S3 バージョニング
- S3 インベントリ
- S3 レプリケーション
- オブジェクトタグ
- S3 Select
- サーバーアクセスログ
- 静的ウェブサイトホスティング
- S3 Storage Lens
- S3 Storage Lens グループ
- S3 Transfer Acceleration
- AWS Key Management Service (AWS KMS) キーによる二層式サーバー側の暗号化 (DSSE-KMS)
- AWS Key Management Service (AWS KMS) キー (SSE-KMS) によるサーバー側の暗号化
- 顧客提供のキーを用いたサーバー側の暗号化 (SSE-C)。
- AWS Management Console で新しいバケットを作成する際に既存のバケット設定をコピーするオプション。

S3 Express One Zone の使用を開始する

次のセクションでは、Amazon S3 Express One Zone ストレージクラスとディレクトリバケットの使用を開始する方法について説明します。詳細については、「S3 Express One Zone とは」を参照してください。

トピック

- S3 Express One Zone で AWS Identity and Access Management (IAM) を設定する
- ゲートウェイ VPC エンドポイントを設定する

- S3 コンソール、AWS CLI、AWS SDK を使用して S3 Express One Zone を使用します。

S3 Express One Zone で AWS Identity and Access Management (IAM) を設定する

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に管理するうえで役立つ AWS のサービス です。IAM 管理者は、どのユーザーが認証 (サインイン) して、どのユーザーにリソースの使用を許可する (アクセス許可を持たせる) かを制御します。IAMは追加料金なしでご利用いただけます。

デフォルトでは、ユーザーにはディレクトリバケットと S3 Express One Zone オペレーションのためのアクセス許可はありません。ディレクトリバケットと S3 Express One Zone オペレーションに対するアクセス許可を付与するには、IAM を使用してユーザーまたはロールを作成し、それらのアイデンティティにアクセス許可をアタッチします。

IAM の使用を開始するには、「S3 Express One Zone 向け AWS Identity and Access Management (IAM)」と「S3 Express One Zone の IAM アイデンティティベースのポリシー」を参照してください。

ゲートウェイ VPC エンドポイントを設定する

S3 Express One Zone にアクセスするには、標準の Amazon S3 エンドポイントとは異なるリージョンエンドポイントとゾーンエンドポイントを使用します。使用する Amazon S3 API オペレーションに応じて、ゾーンエンドポイントまたはリージョンエンドポイントのいずれかが必要です。サポートされている API オペレーションのエンドポイントタイプ別の全リストについては、「S3 Express One Zone がサポートする API オペレーション」を参照してください。ゾーンエンドポイントとリージョンエンドポイントの両方にアクセスするには、ゲートウェイ仮想プライベートクラウド (VPC) エンドポイントを介す必要があります。ゲートウェイエンドポイントを設定するには、「S3 Express One Zone のネットワーク」を参照してください。

S3 コンソール、AWS CLI、AWS SDK を使用して S3 Express One Zone を使用します。

AWS SDK、Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、Amazon S3 REST API を使用して S3 Express One Zone ストレージクラスとディレクトリバケットを操作できます。

S3 コンソール

S3 コンソールを使って使用を開始するには、次の手順に従います。

- ディレクトリバケットの作成
- ディレクトリバケットを空にする
- ディレクトリバケットの削除

AWS SDK

S3 Express One Zone は次の AWS SDK をサポートしています。

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java 2.x
- AWS SDK for JavaScript v3
- AWS SDK for .NET
- AWS SDK for PHP
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby
- AWS SDK for Kotlin
- AWS SDK for Rust

S3 Express One Zone を使用する場合は、最新バージョンの AWS SDK を使用することをお勧めします。S3 Express One Zone でサポートされている AWS SDK は、ユーザーに代わってセッションの確立、更新、終了を処理します。つまり、AWS SDK をダウンロードしてインストールし、必要な IAM アクセス許可を設定すると、直ちに API オペレーションを使用開始できます。詳細については、「S3 Express One Zone 向け AWS Identity and Access Management (IAM)」を参照してください。

AWS SDK のダウンロードやインストールなどの詳細については、「AWS での構築ツール」を参照してください。

AWS SDK の例については、次を参照してください。

- ディレクトリバケットの作成
- ディレクトリバケットを空にする

- ディレクトリバケットの削除

AWS Command Line Interface (AWS CLI)

AWS Command Line Interface (AWS CLI) を使用してディレクトリ バケットを作成し、S3 Express One Zone でサポートされているリージョンとゾーンのエンドポイント API オペレーションを使用できます。

AWS CLI の使用を開始するには、「AWS CLI コマンドレファレンス」の「AWS CLI の使用を開始する」を参照してください。

> **Note**
>
> ディレクトリバケットで高レベルの aws s3 コマンドを使用するには、AWS CLI を最新バージョンに更新します。AWS CLI のインストールと設定方法の詳細については、「AWS CLIコマンドリファレンス」の「AWS CLI の最新バージョンをインストールまたは更新する」を参照してください。

AWS CLI の例については、次を参照してください。

- ディレクトリバケットの作成
- ディレクトリバケットを空にする
- ディレクトリバケットの削除

S3 Express One Zone のネットワーク

Amazon S3 Express One Zone ストレージクラスオブジェクトとディレクトリバケットにアクセスするには、標準の Amazon S3 エンドポイントとは異なるリージョン API エンドポイントとゾーン API エンドポイントを使用します。使用する S3 API オペレーションに応じて、ゾーンエンドポイントまたはリージョンエンドポイントのいずれかが必要です。エンドポイント別の API オペレーションの全リストについては、「S3 Express One Zone がサポートする API オペレーション」を参照してください。

ゲートウェイ仮想プライベートクラウド (VPC) エンドポイントを通じて、ゾーン API オペレーションとリージョン API オペレーションの両方にアクセスできます。ゲートウェイ VPC エンドポイントを設定するには、「the section called “VPC ゲートウェイエンドポイントの設定”」を参照してください。

次のトピックでは、ゲートウェイ VPC エンドポイントを使用して S3 Express One Zone にアクセスするためのネットワーク要件について説明します。

トピック

エンドポイント

ゲートウェイ VPC エンドポイントを使用して、VPC から Amazon S3 Express One Zone ストレージクラスオブジェクトとディレクトリバケットにアクセスできます。S3 Express One Zone はリージョン API エンドポイントとゾーン API エンドポイントを使用します。使用する Amazon S3 API オペレーションに応じて、リージョンゾーンエンドポイントまたはゾーンエンドポイントのいずれかが必要です。ゲートウェイエンドポイントは追加料金なしで使用できます。

バケットレベル (またはコントロールプレーン) API オペレーションは、リージョンエンドポイントを通じて利用でき、リージョンエンドポイント API オペレーションと呼ばれます。リージョンエンドポイント API オペレーションの例には、`CreateBucket`、`DeleteBucket` があります。ディレクトリバケットを作成する際は、ディレクトリバケットを作成する単一のアベイラビリティーゾーンを選択します。ディレクトリバケットを作成した後、ゾーンエンドポイント API オペレーションを使用して、ディレクトリバケット内のオブジェクトをアップロードして管理できます。

オブジェクトレベル (またはデータプレーン) の API オペレーションは、ゾーンエンドポイントを通じて利用でき、ゾーンエンドポイント API オペレーションと呼ばれます。ゾーン別エンドポイント API オペレーションの例には、`CreateSession`、`PutObject` があります。

各リージョンとアベイラビリティーゾーンで使用できるリージョン API エンドポイントとゾーン API エンドポイントは、次の表のとおりです。

VPC ゲートウェイエンドポイントの設定

次の手順を使用して、Amazon S3 Express One Zone ストレージクラスのオブジェクトとディレクトリバケットに接続するゲートウェイエンドポイントを作成します。

ゲートウェイ VPC エンドポイントを設定するには

1. https://console.aws.amazon.com/vpc/ で Amazon VPC コンソールを開きます。
2. ナビゲーションペインで、[エンドポイント] を選択します。

3. [エンドポイントの作成] を選択します。
4. エンドポイントの名前を作成します。
5. [Service category] (サービスカテゴリ) で、AWS のサービス を選択します。
6. [サービス] では、[Type=Gateway] フィルターを追加して、com.amazonaws.***region***.s3express の隣にあるボタンをクリックします。
7. [VPC] で、エンドポイントを作成する VPC を選択します。
8. [Route tables] (ルートテーブル) で、エンドポイントで使用するルートテーブルを選択します。Amazon VPC は、サービスに送信するトラフィックをエンドポイントネットワークインターフェイスにポイントするルートを自動的に追加します。
9. [ポリシー] では、[フルアクセス] を選択して、VPC エンドポイントのすべてのリソースに対するすべてのプリンシパルによるすべてのオペレーションを許可します。それ以外の場合は、[カスタム] を選択して、プリンシパルが VPC エンドポイントを介してリソースに対してアクションを実行するために必要なアクセス許可を制御する VPC エンドポイントポリシーをアタッチします。
10. (オプション) タグを追加するには、[新しいタグを追加] をクリックして、タグのキーバリューを入力します。
11. [エンドポイントの作成] を選択します。

ゲートウェイエンドポイントを作成した後、リージョン API エンドポイントとゾーン API エンドポイントを使用して、Amazon S3 Express One Zone ストレージクラスのオブジェクトとディレクトリバケットにアクセスできます。

ディレクトリバケット

Amazon S3 バケットには、汎用バケットとディレクトリバケットの 2 種類があります。アプリケーションとパフォーマンス要件に最適なバケットタイプを選択します。

- 汎用バケット はオリジナルの S3 バケットタイプであり、ほとんどのユースケースやアクセスパターンに推奨されます。汎用バケットでは、S3 Express One Zone 以外のすべてのストレージクラスにオブジェクトを保存することもできます。
- ディレクトリバケットは、S3 Express One Zone ストレージクラスのみを使用します。アプリケーションがパフォーマンスの影響を受けやすく、1 桁ミリ秒の PUT と GET のレイテンシーから利点が得られる場合にお勧めします。

ディレクトリバケットは一貫して 1 桁ミリ秒のレイテンシーに維持する必要があるワークロードまたはパフォーマンス重視のアプリケーションで使用されます。ディレクトリバケットは、汎用バケットのフラットなストレージ構造とは対照的に、データを階層的にディレクトリにまとめます。ディレクトリバケットにはプレフィックスの制限はなく、個々のディレクトリは水平方向にスケールできます。

ディレクトリバケットは、S3 Express One Zone ストレージクラスを使用します。このストレージクラスでは、単一のアベイラビリティゾーン内の複数のデバイスにわたってデータを保存します。ただし、アベイラビリティゾーン全体でデータを冗長的に保存することはありません。ディレクトリバケットを作成する場合、Amazon EC2、Amazon Elastic Kubernetes Service、または Amazon Elastic Container Service (Amazon ECS) コンピュートインスタンスのローカルにある AWS リージョン とアベイラビリティーゾーンを指定して、パフォーマンスを最適化することをお勧めします。

各 AWS アカウント に最大 10 のディレクトリバケットを作成できます。バケットに保存できるオブジェクトの数に制限はありません。バケット クォータは、AWS アカウント の各リージョンに適用されます。アプリケーションでこの制限を増やす必要がある場合は、AWS Support にお問い合わせください。詳細については、「Service Quotas コンソール」を参照してください。

> **Important**
>
> 少なくとも 90 日間リクエストアクティビティがないディレクトリバケットは、非アクティブ状態に移行します。非アクティブ状態の場合、ディレクトリバケットは一時的に読み取りと書き込みが利用できなくなります。非アクティブなバケットには、ストレージ、オブジェクトメタデータ、バケットメタデータがすべて保持されます。非アクティブなバケットには既存のストレージ料金が適用されます。非アクティブなバケットへのアクセスをリクエストすると、バケットは通常数分以内にアクティブな状態に移行します。この移行期間中に読み取りと書き込みを行うと HTTP 503 (Service Unavailable) エラーコードが返されます。

次のトピックでは、ディレクトリバケットについて説明します。汎用バケットの詳細については、「バケットの概要」を参照してください。

トピック

- アベイラビリティーゾーン
- ディレクトリバケット名
- ディレクトリ

- キー名
- アクセス管理
- ディレクトリバケットの使用
- ディレクトリバケットの命名規則
- ディレクトリバケットの作成
- ディレクトリバケットのプロパティの表示
- ディレクトリバケットのバケットポリシーの管理
- ディレクトリバケットを空にする
- ディレクトリバケットの削除
- ディレクトリバケットの一覧表示
- ディレクトリバケットでの HeadBucket の使用

アベイラビリティーゾーン

ディレクトリバケットを作成する際は、バケットの配置場所であるアベイラビリティーゾーンと AWS リージョン を選択します。

ディレクトリバケットは、パフォーマンス重視のアプリケーションで使用されるように構築された S3 Express One Zone ストレージクラスを使用します。S3 Express One Zone は、オブジェクトストレージをコンピューティングリソースと同じ場所に配置するオプションを提供し、単一のアベイラビリティーゾーンを選択できる最初の S3 ストレージクラスです。これにより、アクセス速度が最大限に高速化します。

S3 Express One Zone を使用すると、データは単一のアベイラビリティーゾーン内の複数のデバイスに冗長的に保存されます。S3 Express One Zone は、単一のアベイラビリティーゾーン内で 99.95% の可用性を実現するように設計されており、Amazon S3 サービスレベル契約に基づいています。詳細については、「単一のアベイラビリティーゾーン」を参照してください。

ディレクトリバケット名

ディレクトリバケット名は、指定するベース名と、バケットが配置されているアベイラビリティゾーンの ID を含むサフィックスで構成されます。ディレクトリバケット名は次の形式を使用し、ディレクトリバケットの命名規則に従う必要があります。

```
bucket-base-name--azid--x-s3
```

例えば、次のディレクトリバケット名にはアベイラビリティーゾーン ID usw2-az1 が含まれています。

```
bucket-base-name--usw2-az1--x-s3
```

詳細については、「ディレクトリバケットの命名規則」を参照してください。

ディレクトリ

ディレクトリバケットは、汎用バケットのフラットなストレージ構造とは対照的に、データを階層的にディレクトリにまとめます。各 S3 ディレクトリバケットは、バケット内のディレクトリの数を問わず、1 秒あたり数十万のトランザクション (TPS) をサポートできます。

階層型名前空間では、オブジェクトキーの区切り文字が重要です。ただし、サポートされている唯一の区切り文字はスラッシュ (/) です。ディレクトリは区切り文字の境界で定義されます。例えば、dir1/dir2/file1.txt オブジェクトキーを使用すると、ディレクトリ dir1/ と dir2/ が自動的に作成され、file1.txt オブジェクトが dir1/dir2/file1.txt パス内の /dir2 ディレクトリに追加されます。

ディレクトリバケットインデックスモデルは、ListObjectsV2 API オペレーションに対して並べ替えされていない結果を返します。結果をバケットのサブセクションに限定する必要がある場合は、prefix=dir1/ など、prefix パラメータにサブディレクトリパスを指定できます。

キー名

ディレクトリバケットの場合、複数のオブジェクト キーに共通のサブディレクトリが最初のオブジェクト キーで作成されます。同じサブディレクトリの追加のオブジェクトキーは、以前に作成したサブディレクトリを使用します。このモデルでは、アプリケーションに最適なオブジェクトキーを柔軟に選択でき、疎ディレクトリと密ディレクトリが同等にサポートされます。

アクセス管理

ディレクトリバケットでは、バケットレベルですべての S3 ブロックパブリックアクセス設定がデフォルトで有効になっています。S3 オブジェクト所有者はバケット所有者の強制に設定され、アクセスコントロールリスト (ACL) が無効になっています。上記の設定は変更できません。

デフォルトでは、ユーザーにはディレクトリバケットと S3 Express One Zone オペレーションのためのアクセス許可はありません。ディレクトリバケットへのアクセス権限を付与するには、IAM を使用してユーザー、グループ、またはロールを作成し、それらのアイデンティティにアクセス許可

をアタッチします。詳細については、「AWS Identity and Access Management (IAM) for S3 Express One Zone」を参照してください。

ディレクトリバケットの使用

ディレクトリバケットのオペレーションの詳細については、以降のトピックを参照してください。

トピック

- ディレクトリバケットの命名規則
- ディレクトリバケットの作成
- ディレクトリバケットのプロパティの表示
- ディレクトリバケットのバケットポリシーの管理
- ディレクトリバケットを空にする
- ディレクトリバケットの削除
- ディレクトリバケットの一覧表示
- ディレクトリバケットでの HeadBucket の使用

ディレクトリバケットの命名規則

Amazon S3 でディレクトリバケットを作成する場合、次のバケット命名規則が適用されます。汎用バケットの命名規則の詳細については、「バケットの名前付け」を参照してください。

ディレクトリバケット名は、指定するベース名と、バケットが配置されている AWS アベイラビリティーゾーンの ID と `--x-s3` を含むサフィックスで構成されます。

```
base-name--azid--x-s3
```

例えば、次のディレクトリバケット名にはアベイラビリティーゾーン ID `usw2-az1` が含まれています。

```
bucket-base-name--usw2-az1--x-s3
```

> **Note**
>
> コンソールを使用してディレクトリバケットを作成すると、指定したベース名にサフィックスが自動的に追加されます。このサフィックスには、選択したアベイラビリティーゾーンのアベイラビリティーゾーン ID が含まれます。

API を使用してディレクトリバケットを作成する場合は、リクエストでアベイラビリティーゾーン ID を含む完全なサフィックスを指定する必要があります。アベイラビリティーゾーン ID のリストについては、「S3 Express One Zone のアベイラビリティーゾーンとリージョン」を参照してください。

ディレクトリバケット名は、次の条件を満たす必要があります。

- 選択した AWS リージョン とアベイラビリティーゾーン内では一意であること。
- 名前は、サフィックスを含め 3 文字 (最短) から 63 文字 (最長) の長さでなければなりません。
- 小文字の英文字、数字、およびハイフン (-) で構成されていること。
- 文字や数字で始まり、文字や数字で終わります。
- `--azid--x-s3` のサフィックスを含める必要があります。

ディレクトリバケットの作成

Amazon S3 Express One Zone ストレージクラスの使用を開始するには、ディレクトリバケットを使用する必要があります。S3 Express One Zone ストレージクラスを使用するには、ディレクトリバケットを使用する必要があります。S3 Express One Zone ストレージ クラスは、低レイテンシーのユースケースをサポートし、単一のアベイラビリティーゾーン内でより高速なデータ処理を提供します。アプリケーションのパフォーマンスが重視され、1 桁のミリ秒単位の PUT と GET のレイテンシーでの利点が得られる場合は、S3 Express One Zone ストレージクラスを使用できるようにディレクトリバケットを作成することをお勧めします。

Amazon S3 バケットには、汎用バケットとディレクトリバケットの 2 種類があります。アプリケーションとパフォーマンス要件に最適なバケットタイプを選択する必要があります。汎用バケットはオリジナルの S3 バケットタイプです。汎用バケットはほとんどのユースケースとアクセスパターンに推奨されます。S3 Express One Zone を除くすべてのストレージクラスにわたってオブジェクトを保存できます。汎用バケットの詳細については、「バケットの概要」を参照してください。

ディレクトリバケットは、S3 Express One Zone ストレージクラスを使用します。これは、一貫して 1 桁ミリ秒のレイテンシーに維持する必要があるワークロードまたはパフォーマンス重視のアプリケーション向けに設計されています。S3 Express One Zone は、オブジェクトストレージをコンピューティングリソースと同じ場所に配置するオプションを提供し、単一のアベイラビリティーゾーンを選択できる最初の S3 ストレージクラスです。これにより、アクセス速度が最大限に高速化します。ディレクトリバケットを作成する場合、必要に応じて Amazon EC2、Amazon Elastic Kubernetes Service、または Amazon Elastic Container Service (Amazon ECS) コンピュートインス

タンスのローカルにある AWS リージョン とアベイラビリティーゾーンを指定して、パフォーマンスを最適化できます。

S3 Express One Zone では、データは単一のアベイラビリティーゾーン内で冗長的に複数のデバイスに保存されます。S3 Express One Zone は、単一のアベイラビリティーゾーン内で 99.95% の可用性を実現するように設計されており、Amazon S3 サービスレベル契約に基づいています。詳細については、「単一のアベイラビリティーゾーン」を参照してください。

ディレクトリバケットは、汎用バケットのフラットなストレージ構造とは対照的に、データを階層的にディレクトリにまとめます。ディレクトリバケットにはプレフィックスの制限はなく、個々のディレクトリは水平方向にスケールできます。

汎用バケットの詳細については、「ディレクトリバケット」を参照してください。

ディレクトリバケット名

ディレクトリバケット名は次の形式を使用し、ディレクトリバケットの命名規則に従う必要があります。

```
bucket-base-name--azid--x-s3
```

例えば、次のディレクトリバケット名にはアベイラビリティーゾーン ID usw2-az1 が含まれています。

```
bucket-base-name--usw2-az1--x-s3
```

ディレクトリバケットの命名規則の詳細については、「ディレクトリバケットの命名規則」を参照してください。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ページ上部にあるナビゲーションバーで、現在表示されている AWS リージョン の名前をクリックします。次に、バケットを作成するリージョンを選択します。

> **Note**
>
> レイテンシーとコストを最小化するため、さらに規制条件に対応するために、最寄りのリージョンを選択します。明示的に別のリージョンに移動する場合を除き、特定のリー

ジョンに保管されたオブジェクトは、そのリージョンから移動されることはありません。Amazon S3 AWS リージョン のリストについては、「Amazon Web Services 全般のリファレンス」の「AWS のサービス エンドポイント」を参照してください。

3. 左側のナビゲーションペインで、[バケット] を選択します。
4. [バケットを作成] を選択します。

   [バケットの作成] ページが開きます。
5. [全般設定] で、バケットが作成される AWS リージョン を確認します。
6. [バケットタイプ] で [ディレクトリ] を選択します。

> **Note**
>
> - ディレクトリバケットをサポートしていないリージョンを選択した場合、[バケットタイプ] オプションは表示されなくなり、バケットタイプはデフォルトで汎用バケットになります。ディレクトリバケットを作成するには、サポートされているリージョンを選択する必要があります。ディレクトリバケットと Amazon S3 Express One Zone ストレージクラスをサポートするリージョンのリストについては、「the section called “S3 Express One Zone のアベイラビリティーゾーンとリージョン”」を参照してください。
> - バケット作成後にバケットタイプは変更できません。

[アベイラビリティーゾーン] では、コンピューティングサービスにローカルなアベイラビリティーゾーンを選択します。ディレクトリバケットと S3 Express One Zone ストレージクラスをサポートするアベイラビリティーゾーンのリストについては、「the section called “S3 Express One Zone のアベイラビリティーゾーンとリージョン”」を参照してください。

> **Note**
>
> このアベイラビリティーゾーンは、バケットの作成後に変更することはできません。

7. [アベイラビリティーゾーン] でチェックボックスをオンにして、アベイラビリティーゾーンに障害が発生した場合に、データが使用できなくなったり、データが失われたりする場合があることに同意します。

> **Important**
>
> ディレクトリバケットは、単一のアベイラビリティーゾーン内の複数のデバイスに保存されますが、ディレクトリバケットはアベイラビリティーゾーン間でデータを冗長に保存することはありません。

8. [バケット名] には、ディレクトリバケット名を入力します。

   ディレクトリバケット名は、次の条件を満たす必要があります。

   - 選択した AWS リージョン とアベイラビリティーゾーン内では一意であること。
   - 名前は、サフィックスを含め 3 文字 (最短) から 63 文字 (最長) の長さでなければなりません。
   - 小文字の英文字、数字、およびハイフン (-) で構成されていること。
   - 文字や数字で始まり、文字や数字で終わります。
   - `--azid--x-s3` のサフィックスを含める必要があります。

   コンソールを使用してディレクトリバケットを作成すると、指定したベース名にサフィックスが自動的に追加されます。このサフィックスには、選択したアベイラビリティーゾーンのアベイラビリティーゾーン ID が含まれます。

   バケットを作成したら、その名前を変更することはできません。バケットの命名の詳細については、「バケットの名前付け」を参照してください。

   > **Important**
   >
   > バケット名にアカウント番号などの機密情報を含めないでください。バケット名は、バケット内のオブジェクトを参照する URL に表示されます。

9. [オブジェクト所有者] で、[バケット所有者の強制] の設定が自動的に有効になり、アクセスコントロールリスト (ACL) が無効になります。ディレクトリバケットの場合、ACL は有効にできません。

   ACL は無効です

   - バケット所有者強制 (デフォルト) – ACL は無効になり、バケット所有者はバケット内のすべてのオブジェクトを自動的に所有し、完全に制御します。ACL は、S3 バケット内のデータに

対するアクセス許可に対して影響を与えません。このバケットはアクセスコントロールを定義するためだけにポリシーを使用します。

Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

10. [このバケットのブロックパブリックアクセス設定] で、ディレクトリバケットのパブリックアクセスブロック設定がすべて自動的に有効になります。ディレクトリバケットの場合、この設定は変更できません。ブロックパブリックアクセスの詳細については、Amazon S3 ストレージへのパブリックアクセスのブロック を参照してください。
11. [サーバー側の暗号化設定] で、Amazon S3 は、すべての S3 バケットの暗号化の基本レベルとして、Amazon S3 マネージドキー (SSE-S3) を使用したサーバー側の暗号化を適用しています。ディレクトリバケットへのオブジェクトのアップロードはすべて SSE-S3 で暗号化されます。ディレクトリバケットの場合、暗号化タイプを変更することはできません。SSE-KMS に関する詳細は、「the section called “Amazon S3 マネージド暗号化キー (SSE-S3)”」を参照してください。
12. [バケットを作成] を選択します。

    バケットを作成したら、ファイルやフォルダをバケットに追加できます。詳細については、「the section called “ディレクトリバケットでのオブジェクトの使用”」を参照してください。

AWS SDK の使用

SDK for Go

AWS SDK for Go を使用してディレクトリバケットを作成する方法は、次の例のとおりです。

Example

```
var bucket = "..."

func runCreateBucket(c *s3.Client) {
    resp, err := c.CreateBucket(context.Background(), &s3.CreateBucketInput{
        Bucket: &bucket,
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            Location: &types.LocationInfo{
                Name: aws.String("usw2-az1"),
                Type: types.LocationTypeAvailabilityZone,
            },
```

```
            Bucket: &types.BucketInfo{
                DataRedundancy: types.DataRedundancySingleAvailabilityZone,
                Type:           types.BucketTypeDirectory,
            },
        },
    })
    var terr *types.BucketAlreadyOwnedByYou
    if errors.As(err, &terr) {
        fmt.Printf("BucketAlreadyOwnedByYou: %s\n", aws.ToString(terr.Message))
        fmt.Printf("noop...\n")
        return
    }
    if err != nil {
        log.Fatal(err)
    }

    fmt.Printf("bucket created at %s\n", aws.ToString(resp.Location))
}
```

SDK for Java 2.x

AWS SDK for Java 2.x を使用してディレクトリバケットを作成する方法は、次の例のとおりです。

Example

```
public static void createBucket(S3Client s3Client, String bucketName) {

    //Bucket name format is {base-bucket-name}--{az-id}--x-s3
    //example: doc-example-bucket--usw2-az1--x-s3 is a valid name for a directory
 bucket created in
    //Region us-west-2, Availability Zone 2

    CreateBucketConfiguration bucketConfiguration =
 CreateBucketConfiguration.builder()
             .location(LocationInfo.builder()
                     .type(LocationType.AVAILABILITY_ZONE)
                     .name("usw2-az1").build()) //this must match the Region and
 Availability Zone in your bucket name
             .bucket(BucketInfo.builder()
                    .type(BucketType.DIRECTORY)
                    .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                    .build()).build();
```

```
    try {

             CreateBucketRequest bucketRequest =
 CreateBucketRequest.builder().bucket(bucketName).createBucketConfiguration(bucketConfigurat
             CreateBucketResponse response = s3Client.createBucket(bucketRequest);
             System.out.println(response);
    }

    catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
         }
    }
```

AWS SDK for JavaScript

AWS SDK for JavaScript を使用してディレクトリバケットを作成する方法は、次の例のとおりです。

Example

```
// file.mjs, run with Node.js v16 or higher
// To use with the preview build, place this in a folder
// inside the preview build directory, such as /aws-sdk-js-v3/workspace/

import { S3 } from "@aws-sdk/client-s3";

const region = "us-east-1";
const zone = "use1-az4";
const suffix = `${zone}--x-s3`;

const s3 = new S3({ region });

const bucketName = `...--${suffix}`;

const createResponse = await s3.createBucket(
    { Bucket: bucketName,
      CreateBucketConfiguration: {Location: {Type: "AvailabilityZone", Name: zone},
      Bucket: { Type: "Directory", DataRedundancy: "SingleAvailabilityZone" }}
    }
   );
```

AWS SDK for .NET

AWS SDK for .NET を使用してディレクトリバケットを作成する方法は、次の例のとおりです。

Example

```
using (var amazonS3Client = new AmazonS3Client())
{
    var putBucketResponse = await amazonS3Client.PutBucketAsync(new PutBucketRequest
    {

       BucketName = "DOC-EXAMPLE-BUCKET--usw2-az1--x-s3",
       PutBucketConfiguration = new PutBucketConfiguration
       {
         BucketInfo = new BucketInfo { DataRedundancy =
 DataRedundancy.SingleAvailabilityZone, Type = BucketType.Directory },
         Location = new LocationInfo { Name = "usw2-az1", Type =
 LocationType.AvailabilityZone }
       }
     }).ConfigureAwait(false);
}
```

SDK for PHP

AWS SDK for PHP を使用してディレクトリバケットを作成する方法は、次の例のとおりです。

Example

```
require 'vendor/autoload.php';

$s3Client = new S3Client([

    'region'      => 'us-east-1',
]);


$result = $s3Client->createBucket([
    'Bucket' => 'doc-example-bucket--use1-az4--x-s3',
    'CreateBucketConfiguration' => [
        'Location' => ['Name'=> 'use1-az4', 'Type'=> 'AvailabilityZone'],
        'Bucket' => ["DataRedundancy" => "SingleAvailabilityZone" ,"Type" =>
 "Directory"]   ],
```

```
]);
```

SDK for Python

AWS SDK for Python (Boto3) を使用してディレクトリバケットを作成する方法は、次の例のとおりです。

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def create_bucket(s3_client, bucket_name, availability_zone):
    '''
    Create a directory bucket in a specified Availability Zone

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to create; for example, 'doc-example-bucket--usw2-
az1--x-s3'
    :param availability_zone: String; Availability Zone ID to create the bucket in,
 for example, 'usw2-az1'
    :return: True if bucket is created, else False
    '''

    try:
        bucket_config = {
                'Location': {
                    'Type': 'AvailabilityZone',
                    'Name': availability_zone
                },
                'Bucket': {
                    'Type': 'Directory',
                    'DataRedundancy': 'SingleAvailabilityZone'
                }
            }
        s3_client.create_bucket(
            Bucket = bucket_name,
            CreateBucketConfiguration = bucket_config
        )
    except ClientError as e:
        logging.error(e)
        return False
```

```
    return True


if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    availability_zone = 'usw2-az1'
    s3_client = boto3.client('s3', region_name = region)
    create_bucket(s3_client, bucket_name, availability_zone)
```

SDK for Ruby

AWS SDK for Ruby を使用してディレクトリバケットを作成する方法は、次の例のとおりです。

Example

```
s3 = Aws::S3::Client.new(region:'us-west-2')
s3.create_bucket(
  bucket: "bucket_base_name--az_id--x-s3",
  create_bucket_configuration: {
    location: { name: 'usw2-az1', type: 'AvailabilityZone' },
    bucket: { data_redundancy: 'SingleAvailabilityZone', type: 'Directory' }
  }
)
```

AWS CLI の使用

AWS CLI を使用してディレクトリバケットを作成する方法は、次の例のとおりです。このコマンドを使用する際は、*###############*を独自の情報に置き換えます。

ディレクトリバケットを作成する際は、設定の詳細を指定して、*bucket-base-name*--*azid*--x-s3 の命名規則を使用する必要があります。

```
aws s3api create-bucket
--bucket bucket-base-name--azid--x-s3
--create-bucket-configuration 'Location={Type=AvailabilityZone,Name=usw2-
az1},Bucket={DataRedundancy=SingleAvailabilityZone,Type=Directory}'
--region us-west-2
```

詳細については、「AWS Command Line Interface」の「create-bucket」を参照してください。

ディレクトリバケットのプロパティの表示

Amazon S3 コンソールを使用して、Amazon S3 ディレクトリバケットのプロパティを表示および設定できます。詳細については、ディレクトリバケットおよびS3 Express One Zone とはを参照してください。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [ディレクトリバケット] タブをクリックします。
4. [ディレクトリバケット] リストで、プロパティを確認するバケットの名前を選択します。
5. プロパティ タブを選択します。
6. [プロパティ] タブでは、バケットの次のプロパティを確認できます。

   - ディレクトリバケットの概要 – バケットの AWS リージョン、アベイラビリティーゾーン、Amazon リソースネーム (ARN)、作成日を確認できます。
   - デフォルトでの暗号化 – Amazon S3 は、すべての S3 バケットの暗号化の基本レベルとして、Amazon S3 マネージドキー (SSE-S3) を使用したサーバー側の暗号化を適用します。ディレクトリバケットの場合、この設定は変更できません。Amazon S3 は、ディスクに保存する前にオブジェクトを暗号化し、ダウンロード時にオブジェクトを復号化します。詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。

     ディレクトリバケットでサポートされている機能の詳細については、「S3 Express One Zone の機能」を参照してください。

ディレクトリバケットのバケットポリシーの管理

Amazon S3 コンソールと AWS SDK を使用して、Amazon S3 ディレクトリバケットのバケットポリシーを追加、削除、更新、表示できます。詳細については、以下のトピックを参照してください。S3 Express One Zone でサポートされている AWS Identity and Access Management (IAM) アクションと条件キーの詳細については、「S3 Express One Zone 向け AWS Identity and Access Management (IAM)」を参照してください。ディレクトリバケットのバケット ポリシーの例については、「S3 Express One Zone のディレクトリバケットポリシーの例」を参照してください。

トピック

バケットポリシーの追加

バケットポリシーをディレクトリバケットに追加するには、Amazon S3 コンソール、AWS SDK、または AWS CLI を使用できます。

S3 コンソールの使用

バケットポリシーを作成または編集するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [ディレクトリバケット] タブをクリックします。
4. [ディレクトリバケット] リストで、フォルダやファイルのアップロード先のバケットの名前を選択します。
5. [アクセス許可] タブを選択します。
6. [バケットポリシー] で [編集] を選択します。[バケットポリシーを編集] ページが表示されます。
7. ポリシーを自動的に生成するには、[Policy Generator] を選択します。

   [Policy Generator] を選択すると、新しいウィンドウで AWS Policy Generator が開きます。

   AWS Policy Generator を使用しない場合は、[ポリシー] セクションで JSON ステートメントを追加または編集できます。

   a. [Select Type of Policy] (ポリシーの種類の選択) の [AWS Policy Generator] ページで、[S3 Bucket Policy] (S3 バケットポリシー)を選択します。

   b. 提供されたフィールドに情報を入力してステートメントを追加し、ステートメントの追加を選択します。このステップを、追加するステートメントの数だけ繰り返します。ポリシーステートメントの詳細については、IAM ユーザーガイドの IAM JSON ポリシーのエレメントのリファレンスを参照してください。

> **Note**
>
> わかりやすいように、[バケットポリシーの編集] ページでは、現在のバケットの [バケット ARN] (Amazon リソースネーム) が [ポリシー] テキストフィールドの上に表示されます。この ARN をコピーして、AWSポリシージェネレータのステートメントで使用できます。

c. ステートメントの追加が完了したら、ポリシーの生成を選択します。

d. 生成されたポリシーテキストをコピーし、[閉じる] を選択すると、Amazon S3 コンソールのバケットポリシーの編集ページに戻ります。

8. [Policy] (ポリシー) ボックスで、既存のポリシーを編集するか、AWS Policy Generator からバケットポリシーを貼り付けます。ポリシーを保存する前に、セキュリティ警告、エラー、一般的な警告、および提案を解決してください。

> **Note**
>
> バケットポリシーのサイズは 20 KB に制限されています。

9. [Save changes] (変更の保存) を選択すると、[Permissions] (アクセス許可) タブへ戻ります。

AWS SDK の使用

SDK for Java 2.x

Example

`PutBucketPolicy` AWS SDK for Java 2.x

```
public static void setBucketPolicy(S3Client s3Client, String bucketName, String
 policyText) {

      //sample policy text
      /**
       * policy_statement = {
       *         'Version': '2012-10-17',
       *         'Statement': [
       *             {
       *                 'Sid': 'AdminPolicy',
       *                 'Effect': 'Allow',
```

```
        *                 'Principal': {
        *                     "AWS": "111122223333"
        *                 },
        *                 'Action': 's3express:*',
        *                 'Resource':
'arn:aws:s3express:region:111122223333:bucket/bucket-base-name--azid--x-s3'
        *             }
        *         ]
        *     }
        */
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();
            s3Client.putBucketPolicy(policyReq);
            System.out.println("Done!");
        }

        catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

AWS CLI の使用

この例では、AWS CLI を使用してバケットポリシーをディレクトリバケットに追加する方法を示します。このコマンドを使用する際は、*###############*を独自の情報に置き換えます。

```
aws s3api put-bucket-policy --bucket bucket-base-name--azid--x-s3 --policy file://
bucket_policy.json
```

bucket_policy.json:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AdminPolicy",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3express*",
            "Resource": "arn:aws:s3express:us-west-2:111122223333:bucket/"
        }
    ]
}
```

詳細については、「AWS Command Line Interface」の「put-bucket-policy」を参照してください。

バケットポリシーの表示

ディレクトリバケットのバケットポリシーを表示するには、次の例を使用します。

AWS CLI の使用

この例では、AWS CLI を使用してディレクトリバケットにアタッチされたバケットポリシーを表示する方法を示します。このコマンドを使用する際は、*###############*を独自の情報に置き換えます。

```
aws s3api get-bucket-policy --bucket bucket-base-name--azid--x-s3
```

詳細については、「AWS Command Line Interface」の「get-bucket-policy」を参照してください。

バケットポリシーを削除する

ディレクトリバケットのバケットポリシーを削除するには、次の例を使用します。

AWS SDK の使用

SDK for Java 2.x

Example

`DeleteBucketPolicy` AWS SDK for Java 2.x

```
public static void deleteBucketPolicy(S3Client s3Client, String bucketName) {
      try {
          DeleteBucketPolicyRequest deleteBucketPolicyRequest =
 DeleteBucketPolicyRequest
                  .builder()
                  .bucket(bucketName)
                  .build()
          s3Client.deleteBucketPolicy(deleteBucketPolicyRequest);
          System.out.println("Successfully deleted bucket policy");
      }

      catch (S3Exception e) {
          System.err.println(e.awsErrorDetails().errorMessage());
          System.exit(1);
      }
```

AWS CLI の使用

この例では、AWS CLI を使用してディレクトリバケットのバケットポリシーを削除する方法を示します。このコマンドを使用する際は、*###############*を独自の情報に置き換えます。

```
aws s3api delete-bucket-policy --bucket bucket-base-name--azid--x-s3
```

詳細については、「AWS Command Line Interface」の「delete-bucket-policy」を参照してください。

ディレクトリバケットを空にする

Amazon S3 コンソールを使用してディレクトリバケットを空にできます。汎用バケットの詳細については、「ディレクトリバケット」を参照してください。

ディレクトリバケットを空にする前に、次の点に注意します。

- ディレクトリバケットを空にすると、すべてのオブジェクトが削除されます。ただし、ディレクトリバケットは保持されます。
- ディレクトリバケットを空にすると、空にするアクションを元に戻すことはできません。
- ディレクトリバケットを空にするアクションの進行中にディレクトリバケットに追加されたオブジェクトは削除される可能性があります。

バケットも削除する場合は、次の点に注意してください。

- バケット自体を削除する前に、ディレクトリバケット内のすべてのオブジェクトを削除する必要があります。
- ディレクトリバケット内で進行中のマルチパートアップロードは、バケット自体を削除する前に中止する必要があります。

> **Note**
>
> AWS Command Line Interface (CLI) による `s3 rm` コマンド、Mountpoint による `delete` オペレーション、AWS Management Console による空のバケットオプションボタンでは、ディレクトリバケット内の進行中のマルチパートアップロードを削除できません。こうした進行中のマルチパートアップロードを削除するには、`ListMultipartUploads` オペレーションを使用してバケット内の進行中のマルチパートアップロードを一覧表示し、`AbortMultupartUpload` オペレーションを使用して進行中のすべてのマルチパートアップロードを中止します。

ディレクトリバケットを削除にする方法については、「ディレクトリバケットの削除」を参照してください。進行中のマルチパートアップロードを中止する方法については、「the section called “マルチパートアップロードの中止”」を参照してください。

汎用バケットを空にする方法については、「バケットを空にする」を参照してください。

S3 コンソールの使用

ディレクトリバケットを空にするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [ディレクトリバケット] タブをクリックします。
4. 空にするバケットの名前の横にあるオプションボタンを選択し、[空にする] を選択します。
5. [バケットを空にする] ページで、テキストフィールドに **`permanently delete`** を入力することでバケットを空にすることを確定し、[空にする] を選択します。
6. [バケットを空にする: ステータス] ページで、バケットを空にするプロセスの進行状況をモニタリングします。

ディレクトリバケットの削除

削除できるのは空の Amazon S3 ディレクトリバケットのみです。ディレクトリバケットを削除する前に、バケット内のすべてのオブジェクトを削除し、進行中のすべてのマルチパートアップロードを中止する必要があります。

ディレクトリバケットを空にする方法については、「ディレクトリバケットを空にする」を参照してください。進行中のマルチパートアップロードを中止する方法については、「the section called “マルチパートアップロードの中止”」を参照してください。

汎用バケットを削除する方法については、「バケットの削除」を参照してください。

S3 コンソールの使用

ディレクトリバケットを空にして進行中のすべてのマルチパートアップロードを中止した後に、バケットを削除できます。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [ディレクトリバケット] タブをクリックします。
4. [ディレクトリバケット] リストで、削除するバケット名の横にあるオプションボタンを選択します。
5. [削除] を選択します。
6. [バケットの削除] ページで、テキストフィールドにバケット名を入力して、バケットの削除を確定します。

   > **Important**
   >
   > ディレクトリバケットを削除すると、元に戻すことはできません。

7. バケットを削除するには、[バケットの削除] をクリックします。

AWS SDK の使用

次の例では、AWS SDK for Java 2.x と AWS SDK for Python (Boto3) を使用してディレクトリバケットを削除します。

SDK for Java 2.x

Example

```
public static void deleteBucket(S3Client s3Client, String bucketName) {

    try {
        DeleteBucketRequest del = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
        s3Client.deleteBucket(del);
        System.out.println("Bucket " + bucketName + " has been deleted");
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

SDK for Python

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def delete_bucket(s3_client, bucket_name):
    '''
    Delete a directory bucket in a specified Region

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to delete; for example, 'doc-example-bucket--usw2-
az1--x-s3'
    :return: True if bucket is deleted, else False
    '''

    try:
        s3_client.delete_bucket(Bucket = bucket_name)
    except ClientError as e:
        logging.error(e)
        return False
    return True
```

```
if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    s3_client = boto3.client('s3', region_name = region)
```

AWS CLI の使用

この例では、AWS CLI を使用してディレクトリバケットを削除する方法を示します。このコマンドを使用する際は、*###############*を独自の情報に置き換えます。

```
aws s3api delete-bucket --bucket bucket-base-name--azid--x-s3 --region us-west-2
```

詳細については、「AWS Command Line Interface」の「delete-bucket」を参照してください。

ディレクトリバケットの一覧表示

次の例は、AWS SDK と AWS CLI を使用してディレクトリバケットをリスト表示する方法を示しています。

AWS SDK の使用

SDK for Java 2.x

Example

次の例では、AWS SDK for Java 2.x を使用してディレクトリバケットを一覧表示します。

```
public static void listBuckets(S3Client s3Client) {
       try {
           ListDirectoryBucketsRequest listDirectoryBucketsRequest =
ListDirectoryBucketsRequest.builder().build();
           ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);
           if (response.hasBuckets()) {
               for (Bucket bucket: response.buckets()) {
                   System.out.println(bucket.name());
                   System.out.println(bucket.creationDate());
               }
           }
       }
```

```
        catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

SDK for Python

Example

次の例では、AWS SDK for Python (Boto3) を使用してディレクトリバケットを一覧表示します。

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_directory_buckets(s3_client):
'''
Prints a list of all directory buckets in a Region

:param s3_client: boto3 S3 client
:return: True if there are buckets in the Region, else False
'''
try:
    response = s3_client.list_directory_buckets()
    for bucket in response['Buckets']:
        print (bucket['Name'])
except ClientError as e:
    logging.error(e)
    return False
return True


if __name__ == '__main__':
    region = 'us-east-1'
    s3_client = boto3.client('s3', region_name = region)
    list_directory_buckets(s3_client)
```

AWS SDK for .NET

Example

次の例では、AWS SDK for .NET を使用してディレクトリバケットを一覧表示します。

```
var listDirectoryBuckets = await amazonS3Client.ListDirectoryBucketsAsync(new
 ListDirectoryBucketsRequest
{
  MaxDirectoryBuckets = 10
  }).ConfigureAwait(false);
```

SDK for PHP

Example

次の例では、AWS SDK for PHP を使用してディレクトリバケットを一覧表示します。

```
require 'vendor/autoload.php';

$s3Client = new S3Client([
    'region'      => 'us-east-1',
]);
$result = $s3Client->listDirectoryBuckets();
```

SDK for Ruby

Example

次の例では、AWS SDK for Ruby を使用してディレクトリバケットを一覧表示します。

```
s3 = Aws::S3::Client.new(region:'us-west-1')
s3.list_directory_buckets
```

AWS CLI の使用

次の `list-directory-buckets` コマンド例は、AWS CLI を使用して *us-east-1* リージョン内のディレクトリバケットをリスト表示する方法を示しています。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3api list-directory-buckets --region us-east-1
```

詳細については、AWS CLI コマンドリファレンスの list-directory-buckets を参照してください。

ディレクトリバケットでの HeadBucket の使用

以下の AWS SDK の例は、HeadBucket API オペレーションを使用して、Amazon S3 ディレクトリバケットが存在し、そのバケットにアクセスする許可があるかどうかを調べる方法を示しています。

AWS SDK の使用

次の AWS SDK for Java 2.x の例は、バケットが存在し、そのバケットにアクセスする許可があるかどうかを調べる方法を示しています。

SDK for Java 2.x

Example

AWS SDK for Java 2.x

```
public static void headBucket(S3Client s3Client, String bucketName) {
   try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest
                .builder()
                .bucket(bucketName)
                .build();
        s3Client.headBucket(headBucketRequest);
        System.out.format("Amazon S3 bucket: \"%s\" found.", bucketName);
   }

   catch (S3Exception e) {
       System.err.println(e.awsErrorDetails().errorMessage());
       System.exit(1);
   }
}
```

AWS CLI の使用

次の head-bucket コマンド例は、AWS CLI を使用して、ディレクトリバケットが存在するか、ユーザーにそのディレクトリバケットへのアクセス許可があるかどうかを判断する方法を示しています。このコマンドを実行するには、ユーザー入力プレースホルダーをユーザー自身の情報に置き換えます。

```
aws s3api head-bucket --bucket bucket-base-name--azid--x-s3
```

詳細については、AWS CLI コマンドリファレンスの head-bucket を参照してください。

ディレクトリバケットでのオブジェクトの使用

Amazon S3 ディレクトリバケット作成後、Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、AWS SDK を使用してオブジェクトを操作できます。

S3 Express One Zone ストレージクラスで保存されているオブジェクトの一括オペレーションの詳細については、「オブジェクト管理」を参照してください。オブジェクトのインポート、アップロード、コピー、削除、ダウンロード、およびディレクトリバケット内のオブジェクトからのメタデータの読み取りの詳細については、以下のトピックを参照してください。

トピック

- ディレクトリバケットへのオブジェクトのインポート
- S3 Express One Zone でのバッチオペレーションの使用
- オブジェクトのディレクトリバケットへのアップロード
- ディレクトリバケットでのマルチパートアップロードの使用
- オブジェクトのディレクトリバケットへのコピー
- ディレクトリバケットのオブジェクトの削除
- ディレクトリバケット内のオブジェクトのダウンロード
- ディレクトリバケットでの HeadObject の使用

ディレクトリバケットへのオブジェクトのインポート

Amazon S3 でディレクトリバケットを作成した後、インポートアクションを使用して新しいバケットにデータを入力できます。インポートは、汎用バケットからディレクトリバケットにオブジェクトをコピーする S3 バッチオペレーションジョブを作成するための効率的な方法です。

> **Note**
>
> インポートジョブには、次の制限が適用されます。
>
> - ソースバケットと送信先バケットの両方が同じ AWS リージョン にあり、同じアカウント内にある必要があります。

- ソースバケットをディレクトリバケットにすることはできません。
- 5 GB を超えるオブジェクトはサポートされていないため、コピーオペレーションから除外されます。
- Glacier Flexible Retrieval、Glacier Deep Archive、Intelligent-Tiering Archive Access 階層、Intelligent-Tiering Deep Archive 階層などのストレージクラス内のオブジェクトは、インポート前に復元する必要があります。
- MD5 チェックサムアルゴリズムを使用してインポートされたオブジェクトは、CRC32 チェックサムを使用するように変換されます。
- インポートされたオブジェクトは、Amazon S3 マネージドキー (SSE-S3) を使用したサーバー側暗号化を使用します。
- インポートされたオブジェクトは、一般的な目的のバケットで使用されるストレージクラスとは別の料金制の Express One Zone ストレージクラスを使用します。多数のオブジェクトをインポートする場合は、このコストの違いを考慮する必要があります。

インポートジョブを設定する際、既存のオブジェクトのコピー元となるソースバケットまたはプレフィックスを指定します。また、ソースオブジェクトにアクセスするための許可を持つ AWS Identity and Access Management (IAM) ロールも指定します。その後 Amazon S3 は、オブジェクトをコピーして、適切なストレージクラスとチェックサム設定を自動的に適用するバッチオペレーションジョブを開始します。

インポートジョブを設定するには、Amazon S3 コンソールを使用します。

Amazon S3 コンソールの使用

オブジェクトをディレクトリバケットにインポートするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで [バケット] を選択してから、[ディレクトリバケット] タブをクリックします。オブジェクトをインポートする先のディレクトリバケットの隣にあるオプションボタンをクリックします。
3. Import (インポート) を選択します。
4. [ソース] には、インポートするオブジェクトを含む汎用バケット (またはプレフィックスを含むバケットパス) を入力します。リストから既存の汎用バケットを選択するには、[Browse S3] をクリックします。

5. [ソースオブジェクトにアクセスおよびコピーするための許可] で、次のいずれかを行い、ソースオブジェクトのインポートに必要なアクセス許可を持つ IAM ロールを指定します。

   - Amazon S3 がユーザーに代わって新しい IAM ロールを作成できるようにするには、[新しい IAM ロールを作成] を選択します。

     > **Note**
     >
     > ソースオブジェクトが AWS Key Management Service (AWS KMS) キーを使用したサーバー側の暗号化 (SSE-KMS) で暗号化されている場合は、[新しい IAM ロールを作成] オプションは選択せず、その代わりに、`kms:Decrypt` アクセス許可を持つ既存の IAM ロールを指定します。
     > Amazon S3 は、このアクセス許可を使用してオブジェクトを復号化します。インポート処理中、Amazon S3 は Amazon S3 マネージドキー (SSE-S3) を使用してサーバー側の暗号化 (SSE-S3) でこれらのオブジェクトを再暗号化します。

   - リストから既存の IAM ロールを選択するには、[既存の IAM ロールから選択] をクリックします。
   - Amazon リソースネーム (ARN) を入力して既存の IAM ロールを指定するには、[IAM ロールの ARN の入力] をクリックして、対応するフィールドに ARN を入力します。

6. [送信先] セクションと [コピーされたオブジェクトの設定] セクションに表示された情報を確認します。[送信先] セクションの情報が適切であれば、[インポート] をクリックしてコピージョブを開始します。

   Amazon S3 コンソールでは、新しいジョブのステータスが [バッチオペレーション] ページに表示されます。ジョブの詳細については、ジョブ名の横にあるオプションボタンを選択して、[アクション] メニューで [詳細を表示] を選択します。オブジェクトのインポート先となるディレクトリバケットを開くには、[インポート先を表示] を選択します。

S3 Express One Zone でのバッチオペレーションの使用

Amazon S3 バッチオペレーションを使用して、S3 バケットに保存されているオブジェクトに対してオペレーションを実行できます。S3 バッチオペレーションの詳細については、「Amazon S3 オブジェクトに対する大規模なバッチオペレーションの実行」を参照してください。

次のトピックでは、ディレクトリバケットで S3 Express One Zone ストレージクラスに保存されているオブジェクトに保存されているオブジェクトに対するバッチオペレーションの実行について説明します。

トピック

ディレクトリバケットでのバッチオペレーションの使用

ディレクトリバケットに保存されているオブジェクトに対して Copy オペレーションと Invoke AWS Lambda 関数 オペレーションを実行できます。Copy を使用すると、(1 つのディレクトリバケットから別のディレクトリバケットへ、など) 同じタイプのバケット間でオブジェクトをコピーできます。汎用バケットとディレクトリバケットの間でコピーすることもできます。Invoke AWS Lambda 関数を使用すると、Lambda 関数を使用して、定義したコードを使用してディレクトリバケット内のオブジェクトに対してアクションを実行できます。

オブジェクトのコピー

同じバケット タイプ間、またはディレクトリバケットと汎用バケット間でコピーできます。ディレクトリバケットにコピーするときは、このバケットタイプに適した Amazon リソースネーム (ARN) 形式を使用する必要があります。ディレクトリバケットの ARN 形式は `arn:aws:s3express:`*`region`*`:`*`account-id`*`:bucket/`*`bucket-base-name`*`--x-s3` です。

S3 コンソールの Import アクションを使用して、ディレクトリバケットにデータを入力することもできます。インポートは、汎用バケットからディレクトリバケットにオブジェクトをコピーするバッチオペレーションジョブを作成するための効率的な方法です。汎用バケットからディレクトリバケットへのインポートコピージョブの場合、S3 はマニフェストを自動的に生成します。詳細については、「ディレクトリバケットへのオブジェクトのインポート」と「マニフェストの指定」を参照してください。

Lambda 関数の呼び出し (**`LambdaInvoke`**)

バッチオペレーションを使用してディレクトリバケットで動作する Lambda 関数を呼び出すには、独自の要件があります。例えば、v2 JSON 呼び出しスキーマを使用して Lambda リクエストを構築し、ジョブを作成する際に InvocationSchemaVersion 2.0 を指定する必要があります。詳細については、「AWS Lambda 関数の呼び出し」を参照してください。

主な違い

バッチオペレーションを使用して、S3 Express One Zone ストレージクラスを使用するディレクトリバケットに保存されたオブジェクトに対して一括操作を実行する場合の主な相違点は、次の一覧のとおりです。

- Amazon S3 は、S3 バケットにコピーされたすべての新しいオブジェクトを自動的に暗号化します。S3 バケットのデフォルトの暗号化設定は常に有効になっており、最低でも Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) に設定されています。ディレクトリバケットでは、sSSE-S3 のみがサポートされています。ディレクトリバケット (ソースまたは送信先) でお客様が提供するキーを使用したサーバー側の暗号化 (SSE-C) または `CopyObject` (AWS Key Management Service) キーを使用したサーバー側の暗号化 (SSE-KMS) を設定する AWS KMS リクエストを行うと、HTTP `400 (Bad Request)` エラーの応答が返されます。
- ディレクトリバケット内のオブジェクトにはタグを付けることができません。指定できるのは空のタグセットのみです。デフォルトでは、バッチオペレーションがタグをコピーします。タグ付きのオブジェクトを汎用バケットからディレクトリバケットにコピーすると、`501 (Not Implemented)` の応答が表示されます。
- S3 Express One Zone では、アップロードまたはダウンロード中にデータを検証するために使用されるチェックサムアルゴリズムを選択するオプションが提供されます。CRC32、CRC32、SHA-1、SHA-256 などのセキュアハッシュアルゴリズム (SHA) や巡回冗長検査 (CRC) データ整合性チェックアルゴリズムのいずれかを選択できます。MD5 ベースのチェックサムは S3 Express One Zone ストレージクラスではサポートされていません。
- デフォルトでは、すべての Amazon S3 バケットで S3 オブジェクト所有者の設定がバケット所有者の強制に指定され、アクセスコントロールリスト (ACL) は無効になります。ディレクトリバケットの場合、この設定は変更できません。オブジェクトは、汎用バケットとディレクトリバケットの間でコピーできます。ただし、ディレクトリバケット間のコピーの場合、デフォルト ACL を上書きすることはできません。
- マニフェストの指定方法を問わず、リスト自体は汎用バケットに保存する必要があります。バッチオペレーションでは、既存のマニフェストのディレクトリバケットからのインポート (や生成されたマニフェストのディレクトリバケットへの保存) はできません。ただし、マニフェスト内に記述されたオブジェクトはディレクトリバケットに保存できます。
- バッチオペレーションでは、S3 インベントリレポート内の口ケーションとしてディレクトリバケットを指定することはできません。インベントリレポートはディレクトリバケットをサポートしていません。`ListObjectsV2` API オペレーションを使用してオブジェクトを一覧表示して、ディレクトリバケット内のオブジェクトのマニフェストファイルを作成できます。その後、CSV ファイルにリストを挿入できます。

アクセス権の付与

コピージョブを実行するには、次のアクセス許可が必要です。

- あるディレクトリバケットから別のディレクトリバケットにオブジェクトをコピーするには、`s3express:CreateSession` アクセス許可が必要です。
- ディレクトリバケットから汎用バケットにオブジェクトをコピーするには、コピー先バケットにオブジェクトコピーを書き込む `s3express:CreateSession` アクセス許可と `s3:PutObject` アクセス許可が必要です。
- 汎用バケットからディレクトリバケットにオブジェクトをコピーするには、`s3express:CreateSession` アクセス許可とコピーされるソースオブジェクトを読み取る `s3:GetObject` アクセス許可が必要です。

  詳細については、「Amazon Simple Storage Service API リファレンス」の「CopyObject」を参照してください。
- Lambda 関数を呼び出すには、Lambda 関数に基づいてリソースにアクセス許可を付与する必要があります。必要となるアクセス許可を判断するには、該当する API アクセス許可を確認します。

オブジェクトのディレクトリバケットへのアップロード

Amazon S3 ディレクトリバケット作成後、オブジェクトをアップロードできます。次の例は、S3 コンソールと AWS SDK を使用してディレクトリバケットにオブジェクトをアップロードする方法を示しています。S3 Express One Zone を使用したオブジェクトの一括アップロードオペレーションの詳細については、「オブジェクト管理」を参照してください。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [ディレクトリバケット] タブをクリックします。
4. フォルダまたはファイルをアップロードするバケットの名前を選択します。
5. [オブジェクト] タブで、[アップロード] を選択します。
6. [アップロード] ページで、以下のいずれかを行います。
   - ファイルとフォルダを点線で示されているアップロードエリアにドラッグアンドドロップします。

- [ファイルの追加] または [フォルダの追加] を選択し、アップロードするファイルまたはフォルダを選択して [開く] または [アップロード] を選択します。

7. [チェックサム] で、使用する [チェックサム関数] を選択します。

   (オプション) サイズが 16 MB 未満の単一のオブジェクトをアップロードする場合は、事前に計算されたチェックサム値を指定することもできます。事前に計算された値を指定すると、Amazon S3 は選択したチェックサム関数を使用して計算した値と比較します。値が一致しない場合、アップロードは開始されません。

8. [アクセス許可] と [プロパティ] セクションのオプションは、自動的にデフォルトに設定され、変更できません。ブロックパブリックアクセスは自動的に有効になり、ディレクトリバケットに対して S3 バージョニングと S3 Object Lock を有効にすることはできません。

   (オプション) オブジェクトにキーと値のペアのメタデータを追加する場合は、[プロパティ] セクションを展開し、[メタデータ]セクションで [メタデータを追加]を選択します。

9. リストされたファイルとフォルダをアップロードするには、[アップロード] を選択します。

   Amazon S3 はオブジェクトとフォルダをアップロードします。アップロードが完了すると、[アップロード: ステータス] ページに成功のメッセージが表示されます。

AWS SDK の使用

SDK for Java 2.x

Example

```
public static void putObject(S3Client s3Client, String bucketName, String objectKey,
 Path filePath) {
       //Using File Path to avoid loading the whole file into memory
       try {
           PutObjectRequest putObj = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   //.metadata(metadata)
                   .build();
           s3Client.putObject(putObj, filePath);
           System.out.println("Successfully placed " + objectKey +" into bucket
 "+bucketName);

       }
```

```
        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
}
```

SDK for Python

Example

```
import boto3
import botocore
from botocore.exceptions import ClientError


def put_object(s3_client, bucket_name, key_name, object_bytes):
    """
    Upload data to a directory bucket.
    :param s3_client: The boto3 S3 client
    :param bucket_name: The bucket that will contain the object
    :param key_name: The key of the object to be uploaded
    :param object_bytes: The data to upload
    """
    try:
        response = s3_client.put_object(Bucket=bucket_name, Key=key_name,
                                Body=object_bytes)
        print(f"Upload object '{key_name}' to bucket '{bucket_name}'.")
        return response
    except ClientError:
        print(f"Couldn't upload object '{key_name}' to bucket '{bucket_name}'.")
        raise

def main():
    # Share the client session with functions and objects to benefit from S3 Express
 One Zone auth key
    s3_client = boto3.client('s3')
    # Directory bucket name must end with --azid--x-s3
    resp = put_object(s3_client, 'doc-bucket-example--use1-az5--x-s3', 'sample.txt',
 b'Hello, World!')
    print(resp)

if __name__ == "__main__":
    main()
```

AWS CLI の使用

次の put-object コマンド例は、AWS CLI を使用して Amazon S3 からオブジェクトをアップロードする方法を示しています。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3api put-object --bucket bucket-base-name--azid--x-s3 --key sampleinut/file001.bin
 --body bucket-seed/file001.bin
```

詳細については、AWS CLI コマンドリファレンスの put-object を参照してください。

ディレクトリバケットでのマルチパートアップロードの使用

マルチパートアップロードプロセスを使用すると、単一のオブジェクトをパートのセットとしてアップロードすることができます。各パートは、オブジェクトのデータの連続する部分です。これらのオブジェクトパートは、任意の順序で個別にアップロードできます。いずれかのパートの送信が失敗すると、他のパートに影響を与えることなくそのパートを再送することができます。オブジェクトのすべてのパートがアップロードされたら、Amazon S3 はこれらのパートを組み立ててオブジェクトを作成します。通常、オブジェクトサイズが 100 MB 以上の場合は、単一のオペレーションでオブジェクトをアップロードする代わりに、マルチパートアップロードを使用することを考慮してください。

マルチパートアップロードの使用には、次の利点があります。

- スループットの向上 – パートを並列にアップロードすることで、スループットを向上させることができます。
- ネットワーク問題からの迅速な回復 – パートサイズが小さいほど、ネットワークエラーが原因で失敗したアップロードを再開する際の影響を最小限に抑えることができます。
- オブジェクトのアップロードの一時停止と再開 – オブジェクトの複数のパートを徐々にアップロードできます。マルチパートアップロードをいったん開始すると、終了期限はありません。マルチパートアップロードを明示的に完了または中止する必要があります。
- オブジェクトの最終的なサイズが不明な状態でアップロードを開始 – オブジェクトの作成中でもアップロードを開始できます。

次の方法でマルチパートアップロードを使用することをお勧めします。

- 安定した高帯域幅ネットワーク経由で大きなオブジェクトをアップロードする場合は、複数スレッドのパフォーマンスのためにオブジェクトパートを並行してアップロードすることで、マルチパートアップロードを使用して利用可能な帯域幅を最大限に活用します。

- むらがあるネットワークでアップロードを実行する場合は、マルチパートアップロードを使用して、アップロードの再開を回避することで、ネットワークエラーに対する回復性を高めます。マルチパートアップロードを使用するときには、アップロード中に中断されたパートのアップロードを再試行するだけで済みます。最初からオブジェクトのアップロードを再開する必要はありません。

マルチパートアップロードを使用してディレクトリバケットの Amazon S3 Express One Zone ストレージクラスにオブジェクトをアップロードする場合、マルチパートアップロードプロセスは、マルチパートアップロードを使用して汎用バケットにオブジェクトをアップロードするプロセスと似ています。ただし、重要な相違点がいくつかあります。

マルチパートアップロードを使用して S3 Express One Zone ゾーンにオブジェクトをアップロードする方法の詳細については、以降のトピックを参照してください。

トピック

マルチパートアップロードのプロセス

マルチパートアップロードは次の 3 ステップのプロセスです。

- アップロードを開始します。
- オブジェクトパートをアップロードします。
- すべてのパートをアップロードしたら、マルチパートアップロードを完了します。

Amazon S3 の側では、マルチパートアップロードの完了リクエストを受け取ると同時に、アップロードされたパートからオブジェクトを構築します。構築されたオブジェクトへは、バケット内の他のオブジェクトと同じようにアクセスできます。

マルチパートアップロードの開始

リクエストを送信すると、アップロード ID を含むレスポンスが Amazon S3 から返されます。アップロード ID はマルチパートアップロードの一意の識別子です。パートのアップロード、パートのリスト、アップロードの完了、アップロードの中止を行うときは常に、このアップロード ID を指定する必要があります。

パートのアップロード

パートをアップロードするときは、アップロード ID に加えて、パート番号を指定する必要があります。S3 Express One Zone でマルチパートアップロードを使用する場合、マルチパート番号は連続するパート番号である必要があります。連続しないパート番号を使用してマルチパートアップロードリクエストを実行しようとすると、`HTTP 400 Bad Request` (無効なパート順序) エラーが生成されます。

パート番号によって、アップロードするオブジェクトに含まれるパートとその位置が一意に識別されます。以前にアップロードしたパートと同じパート番号を使って新しいパートをアップロードした場合、以前のパートは上書きされます。

パートをアップロードするたびに、Amazon S3 は エンティティタグ (ETag) ヘッダーを含むレスポンスを返します。パートのアップロードごとに、パート番号と ETag 値を記録する必要があります。すべてのオブジェクトパートアップロードの ETag 値は同じままですが、パートごとに異なるパート番号が割り当てられます。マルチパートアップロードを完了するためには、残りのリクエストにこれらの値を含める必要があります。

Amazon S3 は、S3 バケットにコピーされたすべての新しいオブジェクトを自動的に暗号化します。マルチパートアップロードを実行する際に、リクエストで暗号化情報を指定しないと、アップロードされたパートの暗号化設定は、送信先バケットのデフォルトの暗号化設定に設定されます。Amazon S3 バケットのデフォルトの暗号化設定は常に有効になっており、最低でも Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) に設定されています。ディレクトリバケットでは、SSE-S3 のみがサポートされています。詳細については、「Amazon S3 マネージドキーを用いたサーバー側の暗号化 (SSE-S3)」を参照してください。

マルチパートアップロードの完了

マルチパートアップロードを完了すると、パート番号に基づいて昇順に連結されたオブジェクトが Amazon S3 によって作成されます。完了リクエストが正常に処理されると、個々のパートはなくなります。

「マルチパートアップロードの完了」リクエストには、アップロード ID と、パート番号およびそれに対応する ETag 値の両方のリストが含まれている必要があります。Amazon S3 からのレスポンス

には、結合されるオブジェクトデータを一意に識別する ETag が含まれます。この ETag は、オブジェクトデータの MD5 ハッシュであるとは限りません。

マルチパートアップロードのリスト化

特定のマルチパートアップロードのパートや、進行中のすべてのマルチパートアップロードをリスト表示できます。パートのリストオペレーションでは、特定のマルチパートアップロードについて既にアップロードしたパートの情報が返されます。パートのリストリクエストを送信するたびに、指定したマルチパートアップロードのパート情報 (最大で1,000個のパート) が Amazon S3 から返されます。マルチパートアップロードに 1,000 を超えるパートが含まれる場合、すべてのパートを取得するにはパートのリストリクエストを追加で送信する必要があります。

返されるパートのリストには、アップロードが完了していないパートは含まれていません。マルチパートアップロードの一覧表示オペレーションを使用すると、進行中のマルチパートアップロードのリストを取得できます。

進行中のマルチパートアップロードとは、開始されているものの、まだ完了または中止されていないアップロードを意味します。各リクエストに最大 1,000 個のマルチパートアップロードが返されます。進行中のマルチパートアップロードが 1,000 個を超える場合、残りのマルチパートアップロードを取得するには、リクエストを追加で送信する必要があります。返されたリストは確認の目的でのみ使用してください。マルチパートアップロードの完了リクエストを送信するときに、このリストの結果を使用しないでください。代わりに、パートのアップロード時に指定したパート番号と、それに対応する、Amazon S3 から返される ETag 値の独自のリストを維持しておいてください。

マルチパートアップロードの一覧の詳細については、「Amazon Simple Storage Service API リファレンス」の「ListParts」を参照してください。

マルチパートアップロードオペレーションを使用したチェックサム

オブジェクトをアップロードする際、オブジェクトの整合性を確認するためのチェックサムアルゴリズムを指定できます。MD5 はディレクトリバケットではサポートされていません。次のセキュアハッシュアルゴリズム (SHA) または巡回冗長検査 (CRC) のデータ整合性チェックアルゴリズムのいずれかを指定できます。

- CRC32
- CRC32C
- SHA-1
- SHA-256

GetObject または HeadObject を使用することで、Amazon S3 REST API または AWS SDK を使用して、個々のパートのチェックサム値を取得できます。まだ処理中のマルチパートアップロードの個々の部分のチェックサム値を取得したい場合は、ListParts を使用できます。

> **Important**
>
> 前述のチェックサムアルゴリズムを使用する場合、マルチパートパーツ番号には連続したパーツ番号を使用する必要があります。連続しないパート番号を使用してマルチパートアップロードリクエストを実行しようとすると、Amazon S3 により HTTP 400 Bad Request (無効なパート順序) エラーが生成されます。

マルチパートオブジェクトでのチェックサムの動作の詳細については、「オブジェクトの整合性をチェックする」を参照してください。

マルチパートアップロードの同時オペレーション

分散開発環境においては、アプリケーションから同じオブジェクトに対して複数の更新が同時に開始されることもありえます。例えば、アプリケーションが同じオブジェクトキーを使用して複数のマルチパートアップロードを開始する場合があります。そのようなアップロードごとに、アプリケーションからパートのアップロードが行われ、アップロードの完了リクエストが Amazon S3 に送信されて、オブジェクトが作成されます。S3 Express One Zone の場合、オブジェクトの作成時間はマルチパートアップロードの完了日となります。

> **Important**
>
> ディレクトリバケットに保存されているオブジェクトのバージョニングはサポートされていません。

マルチパートアップロードと料金

マルチパートアップロードを開始すると、アップロードを完了または中止するまですべてのパートが Amazon S3 によって保持されます。マルチパートアップロードの実行期間を通して、アップロードとそれに関連するパートのために使用されるすべてのストレージ、帯域幅、リクエストに対して課金が行われます。マルチパートアップロードを中止した場合、Amazon S3 はアップロードしたアーティファクトやアップロードしたすべてのパートを削除するため、これらについて課金されることはなくなります。指定されたストレージクラスに関係なく、不完全なマルチパートアップロードの削除

に伴う早期削除料金はありません。料金に関する詳細については、[Amazon S3 の料金] を参照してください。

> **Important**
>
> 完全なマルチパートアップロードリクエストが正常に送信されない場合、オブジェクトパーツは組み立てられず、オブジェクトも作成されません。アップロードされたパートに関連のあるすべてのストレージに対して料金が請求されます。そのため、マルチパートアップロードを完了してオブジェクトを作成するか、マルチパートアップロードを停止してアップロードされたパートを削除することが重要です。
> ディレクトリバケットを削除する前に、進行中のすべてのマルチパートアップロードを完了または中止する必要があります。ディレクトリバケットは S3 ライフサイクル設定をサポートしていません。必要に応じて、アクティブなマルチパートアップロードを一覧表示して、アップロードを中止し、バケットを削除できます。

マルチパートアップロード API オペレーションとアクセス許可

ディレクトリバケットのオブジェクト管理 API オペレーションにアクセスできるようにするには、バケットポリシーまたは AWS Identity and Access Management (IAM) ID ベースのポリシーで `s3express:CreateSession` アクセス許可を付与します。

マルチパートアップロードオペレーションを使用するには、必要なアクセス権限を有している必要があります。バケットポリシーまたは IAM ID ベースのポリシーを使用して、これらのオペレーションを実行するアクセス許可を IAM プリンシパルに付与できます。さまざまなマルチパートアップロードオペレーションに必要なアクセス許可の一覧は、次の表のとおりです。

`Initiator` 要素を使用して、マルチパートアップロードの開始点を特定できます。イニシエータが AWS アカウント である場合、このエレメントは `Owner` エレメントと同じ情報を提供します。イニシエータが IAM ユーザーである場合、このエレメントはユーザー ARN と表示名を提供します。

アクション	必要なアクセス許可
マルチパートアップロードの作成	マルチパートアップロードを作成するには、ディレクトリバケットに対して `s3express:CreateSession` アクションを実行するための許可が必要です。

アクション	必要なアクセス許可
マルチパートアップロードの開始	マルチパートアップロードを開始するには、ディレクトリバケットに対して s3express:CreateSession アクションを実行するための許可が必要です。
パートのアップロード	パートをアップロードするには、ディレクトリバケットに対して s3express:CreateSession アクションを実行するための許可が必要です。 バケット所有者は、イニシエータがパートをアップロードできるように、ディレクトリバケットに対して s3express:CreateSession アクションを実行するための許可をイニシエータに付与する必要があります。
パートのアップロード (コピー)	パートをアップロードするには、ディレクトリバケットに対して s3express:CreateSession アクションを実行するための許可が必要です。 開始者がオブジェクトのパートをアップロードするには、バケット所有者が、開始者にそのオブジェクトでの s3express:CreateSession アクションの実行を許可する必要があります。
マルチパートアップロードを完了する	マルチパートアップロードを完了するには、ディレクトリバケットで s3express:CreateSession アクションを実行するための許可が必要です。 イニシエータがマルチパートアップロードを実行するには、バケット所有者がイニシエータにオブジェクトに対する s3express:CreateSession アクションの実行を許可する必要があります。
マルチパートアップロードの中止	マルチパートアップロードを中止するには、s3express:CreateSession アクションを実行するための許可が必要です。 イニシエータはマルチパートアップロードを中止できるように、s3express:CreateSession アクションを実行するためのアクセス許可を明示的に付与される必要があります。
パートの一覧表示	マルチパートアップロードでパートを一覧表示するには、ディレクトリバケットに対して s3express:CreateSession アクションを実行するための許可が必要です。

アクション	必要なアクセス許可
進行中のマルチパートアップロードの一覧表示	バケットへの進行中のマルチパートアップロードを一覧表示するには、そのバケットに対して `s3:ListBucketMultipartUploads` アクションを実行するための許可が必要です。

マルチパートアップロードに対する API オペレーションサポート

Amazon Simple Storage Service API リファレンスの以下のセクションでは、マルチパートアップロードの Amazon S3 REST API オペレーションについて説明しています。

- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- AbortMultipartUpload
- ListParts
- ListMultipartUploads

例

マルチパートアップロードを使用してディレクトリバケット内の S3 Express One Zone にオブジェクトをアップロードするには、次の例を参照してください。

トピック

- マルチパートアップロードの作成
- マルチパートアップロードのパートのアップロード
- マルチパートアップロードの完了
- マルチパートアップロードの中止
- マルチパートアップロードのコピーオペレーションの作成
- 進行中のマルチパートアップロードの一覧表示
- マルチパートアップロードパートの一覧表示

マルチパートアップロードの作成

次の例は、マルチパートアップロードの作成方法を示しています。

AWS SDK の使用

SDK for Java 2.x

Example

```
/**
 * This method creates a multipart upload request that generates a unique upload ID
 that is used to track
 * all the upload parts
 *
 * @param s3
 * @param bucketName - for example, 'doc-example-bucket--use1-az4--x-s3'
 * @param key
 * @return
 */
 private static String createMultipartUpload(S3Client s3, String bucketName, String
 key) {

     CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
             .bucket(bucketName)
             .key(key)
             .build();

     String uploadId = null;

     try {
         CreateMultipartUploadResponse response =
 s3.createMultipartUpload(createMultipartUploadRequest);
         uploadId = response.uploadId();
     }
     catch (S3Exception e) {
         System.err.println(e.awsErrorDetails().errorMessage());
         System.exit(1);
     }
     return uploadId;
```

SDK for Python

Example

```
def create_multipart_upload(s3_client, bucket_name, key_name):
    '''
   Create a multipart upload to a directory bucket

   :param s3_client: boto3 S3 client
   :param bucket_name: The destination bucket for the multipart upload
   :param key_name: The key name for the object to be uploaded
   :return: The UploadId for the multipart upload if created successfully, else None
   '''

   try:
        mpu = s3_client.create_multipart_upload(Bucket = bucket_name, Key =
 key_name)
        return mpu['UploadId']
    except ClientError as e:
        logging.error(e)
        return None
```

AWS CLI の使用

この例では、AWS CLI を使用してディレクトリバケットへのマルチパートアップロードを作成する方法を示します。このコマンドは、オブジェクト *KEY_NAME* のディレクトリバケット *bucket-base-name*--*azid*--x-s3 へのマルチパートアップロードを開始します。このコマンドを使用する際は、*###############*を独自の情報に置き換えます。

```
aws s3api create-multipart-upload --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
```

詳細については、「AWS Command Line Interface」の「create-multipart-upload」を参照してください。

マルチパートアップロードのパートのアップロード

次の例は、マルチパートアップロードの一部をアップロードする方法を示しています。

AWS SDK の使用

SDK for Java 2.x

次の例は、SDK for Java 2.x を使用して、単一のオブジェクトを複数の部分に分割し、それらの部分をディレクトリバケットにアップロードする方法を示しています。

Example

```
/**
 * This method creates part requests and uploads individual parts to S3 and then
 returns all the completed parts
 *
 * @param s3
 * @param bucketName
 * @param key
 * @param uploadId
 * @throws IOException
 */
 private static ListCompletedPartmultipartUpload(S3Client s3, String bucketName,
 String key, String uploadId, String filePath) throws IOException {

        int partNumber = 1;
        ListCompletedPart completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        // read the local file, breakdown into chunks and process
        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            int position = 0;
            while (position < fileSize) {
                file.seek(position);
                int read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
 buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3.uploadPart(
```

```
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        }

        catch (IOException e) {
            throw e;
        }
        return completedParts;
    }
```

SDK for Python

次の例は、SDK for Python を使用して単一のオブジェクトを複数の部分に分割し、それらの部分をディレクトリバケットにアップロードする方法を示しています。

Example

```
def multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_size):
    '''
    Break up a file into multiple parts and upload those parts to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name for object to be uploaded and for the local file
 that's being uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_size: The size parts that the object will be broken into, in bytes.
                      Minimum 5 MiB, Maximum 5 GiB. There is no minimum size for the
 last part of your multipart upload.
    :return: part_list for the multipart upload if all parts are uploaded
 successfully, else None
    '''
```

```
    part_list = []
    try:
        with open(key_name, 'rb') as file:
            part_counter = 1
            while True:
                file_part = file.read(part_size)
                if not len(file_part):
                    break
                upload_part = s3_client.upload_part(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = mpu_id,
                    Body = file_part,
                    PartNumber = part_counter
                )
                part_list.append({'PartNumber': part_counter, 'ETag':
upload_part['ETag']})
                part_counter += 1
    except ClientError as e:
        logging.error(e)
        return None
    return part_list
```

AWS CLI の使用

この例では、AWS CLI を使用して単一のオブジェクトを複数の部分に分割し、それらの部分をディレクトリバケットにアップロードする方法を示しています。このコマンドを使用する際は、*######### ########*を独自の情報に置き換えます。

```
aws s3api upload-part --bucket bucket-base-name--azid--x-s3 --
key KEY_NAME --part-number 1 --body LOCAL_FILE_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

詳細については、「AWS Command Line Interface」の「upload-part」を参照してください。

マルチパートアップロードの完了

次の例は、マルチパートアップロードを完了する方法を示しています。

AWS SDK の使用

SDK for Java 2.x

次の例は、SDK for Java 2.x を使用してマルチパートアップロードを完了する方法を示しています。

Example

```
/**
 * This method completes the multipart upload request by collating all the upload
 parts
 * @param s3
 * @param bucketName - for example, 'doc-example-bucket--usw2-az1--x-s3'
 * @param key
 * @param uploadId
 * @param uploadParts
 */
 private static void completeMultipartUpload(S3Client s3, String bucketName, String
 key, String uploadId, ListCompletedPart uploadParts) {
        CompletedMultipartUpload completedMultipartUpload =
 CompletedMultipartUpload.builder()
                .parts(uploadParts)
                .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
                CompleteMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .multipartUpload(completedMultipartUpload)
                        .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    public static void multipartUploadTest(S3Client s3, String bucketName, String
 key, String localFilePath)  {
        System.out.println("Starting multipart upload for: " + key);
        try {
            String uploadId = createMultipartUpload(s3, bucketName, key);
            System.out.println(uploadId);
            ListCompletedPart parts = multipartUpload(s3, bucketName, key, uploadId,
 localFilePath);
```

```
            completeMultipartUpload(s3, bucketName, key, uploadId, parts);
            System.out.println("Multipart upload completed for: " + key);
        }

        catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python

次の例は、SDK for Python を使用してマルチパートアップロードを完了する方法を示しています。

Example

```
def complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_list):
    '''
    Completes a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: The destination bucket for the multipart upload
    :param key_name: The key name for the object to be uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_list: The list of uploaded part numbers with their associated ETags
    :return: True if the multipart upload was completed successfully, else False
    '''

    try:
        s3_client.complete_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = mpu_id,
            MultipartUpload = {
                'Parts': part_list
            }
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
```

```
    MB = 1024 ** 2
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'OBJECT_NAME'
    part_size = 10 * MB
    s3_client = boto3.client('s3', region_name = region)
    mpu_id = create_multipart_upload(s3_client, bucket_name, key_name)
    if mpu_id is not None:
        part_list = multipart_upload(s3_client, bucket_name, key_name, mpu_id,
part_size)
        if part_list is not None:
            if complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id,
part_list):
                print (f'{key_name} successfully uploaded through a ultipart upload
to {bucket_name}')
            else:
                print (f'Could not upload {key_name} hrough a multipart upload to
{bucket_name}')
```

AWS CLI の使用

この例では、AWS CLI を使用してディレクトリバケットのマルチパートアップロードを完了する方法を示します。このコマンドを使用する際は、*###############*を独自の情報に置き換えます。

```
aws s3api complete-multipart-upload --bucket bucket-base-name--azid--x-s3 --
key KEY_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
 --multipart-upload file://parts.json
```

この例では、完全なファイルに再構成する必要があるマルチパートアップロードの部分を記述した JSON 構造を採用しています。この例では、`file://` プレフィックスを使用して、`parts` という名前のローカルフォルダーにあるファイルから JSON 構造を読み込みます。

parts.json:

```
parts.json
{
  "Parts": [
    {
      "ETag": "6b78c4a64dd641a58dac8d9258b88147",
      "PartNumber": 1
    }
```

```
  ]
}
```

詳細については、「AWS Command Line Interface」の「complete-multipart-upload」を参照してください。

マルチパートアップロードの中止

次の例は、マルチパートアップロードを中止する方法を示しています。

AWS SDK の使用

SDK for Java 2.x

次の例は、SDK for Java 2.x を使用してマルチパートアップロードを中止する方法を示しています。

Example

```
public static void abortMultiPartUploads( S3Client s3, String bucketName ) {

        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            ListMultipartUpload uploads = response.uploads();

            AbortMultipartUploadRequest abortMultipartUploadRequest;
            for (MultipartUpload upload: uploads) {
                abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(upload.key())
                        .uploadId(upload.uploadId())
                        .build();

                s3.abortMultipartUpload(abortMultipartUploadRequest);
            }

        }
```

```
        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python

次の例は、SDK for Python を使用してマルチパートアップロードを中止する方法を示しています。

Example

```
import logging
import boto3
from botocore.exceptions import ClientError


def abort_multipart_upload(s3_client, bucket_name, key_name, upload_id):
    '''
    Aborts a partial multipart upload in a directory bucket.

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket where the multipart upload was initiated - for
 example, 'doc-example-bucket--usw2-az1--x-s3'
    :param key_name: Name of the object for which the multipart upload needs to be
 aborted
    :param upload_id: Multipart upload ID for the multipart upload to be aborted
    :return: True if the multipart upload was successfully aborted, False if not
    '''
    try:
        s3_client.abort_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = upload_id
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True


if __name__ == '__main__':
    region = 'us-west-2'
```

```
    bucket_name = 'BUCKET_NAME'
    key_name = 'KEY_NAME'
        upload_id = 'UPLOAD_ID'
    s3_client = boto3.client('s3', region_name = region)
    if abort_multipart_upload(s3_client, bucket_name, key_name, upload_id):
        print (f'Multipart upload for object {key_name} in {bucket_name} bucket has
been aborted')
    else:
        print (f'Unable to abort multipart upload for object {key_name} in
{bucket_name} bucket')
```

AWS CLI の使用

次の例は、AWS CLI を使用してマルチパートアップロードを中止する方法を示しています。このコマンドを使用する際は、*###############*を独自の情報に置き換えます。

```
aws s3api abort-multipart-upload --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
 --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
MAQAAAAB0OxUFeA7LTbWWFS8WYwhrxDxTIDN-pdEEq_agIHqsbg"
```

詳細については、「AWS Command Line Interface」の「abort-multipart-upload」を参照してください。

マルチパートアップロードのコピーオペレーションの作成

次の例は、マルチパートアップロードを使用して、あるバケットから別のバケットにオブジェクトをコピーする方法を示しています。

AWS SDK の使用

SDK for Java 2.x

次の例は、SDK for Java 2.x を使用して、マルチパートアップロードを使って、あるバケットから別のバケットにオブジェクトをプログラムでコピーする方法を示しています。

Example

```
/**
 * This method creates a multipart upload request that generates a unique upload ID
 that is used to track
 * all the upload parts.
```

```
*
* @param s3
* @param bucketName
* @param key
* @return
*/
private static String createMultipartUpload(S3Client s3, String bucketName, String
key) {
       CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
               .bucket(bucketName)
               .key(key)
               .build();
       String uploadId = null;
       try {
           CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
           uploadId = response.uploadId();
       } catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return uploadId;
 }

 /**
  * Creates copy parts based on source object size and copies over individual parts
  *
  * @param s3
  * @param sourceBucket
  * @param sourceKey
  * @param destnBucket
  * @param destnKey
  * @param uploadId
  * @return
  * @throws IOException
  */
   public static ListCompletedPart multipartUploadCopy(S3Client s3, String
sourceBucket, String sourceKey, String destnBucket, String destnKey, String
uploadId) throws IOException {

       // Get the object size to track the end of the copy operation.
       HeadObjectRequest headObjectRequest = HeadObjectRequest
               .builder()
```

```
                .bucket(sourceBucket)
                .key(sourceKey)
                .build();
        HeadObjectResponse response = s3.headObject(headObjectRequest);
        Long objectSize = response.contentLength();

        System.out.println("Source Object size: " + objectSize);

        // Copy the object using 20 MB parts.
        long partSize = 20 * 1024 * 1024;
        long bytePosition = 0;
        int partNum = 1;
        ListCompletedPart completedParts = new ArrayList<>();
        while (bytePosition < objectSize) {
            // The last part might be smaller than partSize, so check to make sure
            // that lastByte isn't beyond the end of the object.
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

            System.out.println("part no: " + partNum + ", bytePosition: " +
bytePosition + ", lastByte: " + lastByte);

            // Copy this part.
            UploadPartCopyRequest req = UploadPartCopyRequest.builder()
                    .uploadId(uploadId)
                    .sourceBucket(sourceBucket)
                    .sourceKey(sourceKey)
                    .destinationBucket(destnBucket)
                    .destinationKey(destnKey)
                    .copySourceRange("bytes="+bytePosition+"-"+lastByte)
                    .partNumber(partNum)
                    .build();
            UploadPartCopyResponse res = s3.uploadPartCopy(req);
            CompletedPart part = CompletedPart.builder()
                    .partNumber(partNum)
                    .eTag(res.copyPartResult().eTag())
                    .build();
            completedParts.add(part);
            partNum++;
            bytePosition += partSize;
        }
        return completedParts;
    }
```

```
    public static void multipartCopyUploadTest(S3Client s3, String srcBucket, String
 srcKey, String destnBucket, String destnKey)  {
        System.out.println("Starting multipart copy for: " + srcKey);
        try {
            String uploadId = createMultipartUpload(s3, destnBucket, destnKey);
            System.out.println(uploadId);
            ListCompletedPart parts = multipartUploadCopy(s3, srcBucket,
 srcKey,destnBucket,  destnKey, uploadId);
            completeMultipartUpload(s3, destnBucket, destnKey, uploadId, parts);
            System.out.println("Multipart copy completed for: " + srcKey);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python

次の例は、SDK for Python を使用して、マルチパートアップロードを使って、あるバケットから別のバケットにオブジェクトをプログラムでコピーする方法を示しています。

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def head_object(s3_client, bucket_name, key_name):
    '''
    Returns metadata for an object in a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that contains the object to query for metadata
    :param key_name: Key name to query for metadata
    :return: Metadata for the specified object if successful, else None
    '''

    try:
        response = s3_client.head_object(
            Bucket = bucket_name,
            Key = key_name
        )
        return response
    except ClientError as e:
```

```
        logging.error(e)
        return None

def create_multipart_upload(s3_client, bucket_name, key_name):
    '''
    Create a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name of the object to be uploaded
    :return: UploadId for the multipart upload if created successfully, else None
    '''

    try:
        mpu = s3_client.create_multipart_upload(Bucket = bucket_name, Key =
 key_name)
        return mpu['UploadId']
    except ClientError as e:
        logging.error(e)
        return None

def multipart_copy_upload(s3_client, source_bucket_name, key_name,
 target_bucket_name, mpu_id, part_size):
    '''
    Copy an object in a directory bucket to another bucket in multiple parts of a
 specified size

    :param s3_client: boto3 S3 client
    :param source_bucket_name: Bucket where the source object exists
    :param key_name: Key name of the object to be copied
    :param target_bucket_name: Destination bucket for copied object
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_size: The size parts that the object will be broken into, in bytes.
                      Minimum 5 MiB, Maximum 5 GiB. There is no minimum size for the
 last part of your multipart upload.
    :return: part_list for the multipart copy if all parts are copied successfully,
 else None
    '''

    part_list = []
    copy_source = {
        'Bucket': source_bucket_name,
        'Key': key_name
    }
```

```
    try:
        part_counter = 1
        object_size = head_object(s3_client, source_bucket_name, key_name)
        if object_size is not None:
            object_size = object_size['ContentLength']
        while (part_counter - 1) * part_size <object_size:
            bytes_start = (part_counter - 1) * part_size
            bytes_end = (part_counter * part_size) - 1
            upload_copy_part = s3_client.upload_part_copy (
                Bucket = target_bucket_name,
                CopySource = copy_source,
                CopySourceRange = f'bytes={bytes_start}-{bytes_end}',
                Key = key_name,
                PartNumber = part_counter,
                UploadId = mpu_id
            )
            part_list.append({'PartNumber': part_counter, 'ETag':
 upload_copy_part['CopyPartResult']['ETag']})
            part_counter += 1
    except ClientError as e:
        logging.error(e)
        return None
    return part_list

def complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_list):
    '''
    Completes a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name of the object to be uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_list: List of uploaded part numbers with associated ETags
    :return: True if the multipart upload was completed successfully, else False
    '''

    try:
        s3_client.complete_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = mpu_id,
            MultipartUpload = {
                'Parts': part_list
            }
```

```
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    MB = 1024 ** 2
    region = 'us-west-2'
    source_bucket_name = 'SOURCE_BUCKET_NAME'
    target_bucket_name = 'TARGET_BUCKET_NAME'
    key_name = 'KEY_NAME'
    part_size = 10 * MB
    s3_client = boto3.client('s3', region_name = region)
    mpu_id = create_multipart_upload(s3_client, target_bucket_name, key_name)
    if mpu_id is not None:
        part_list = multipart_copy_upload(s3_client, source_bucket_name, key_name,
 target_bucket_name, mpu_id, part_size)
        if part_list is not None:
            if complete_multipart_upload(s3_client, target_bucket_name, key_name,
 mpu_id, part_list):
                print (f'{key_name} successfully copied through multipart copy from
 {source_bucket_name} to {target_bucket_name}')
            else:
                print (f'Could not copy {key_name} through multipart copy from
 {source_bucket_name} to {target_bucket_name}')
```

AWS CLI の使用

次の例は、AWS CLI を使用して、マルチパートアップロードを使って、あるバケットからあるディレクトリバケットにオブジェクトをプログラムでコピーする方法を示しています。このコマンドを使用する際は、*###############*を独自の情報に置き換えます。

```
aws s3api upload-part-copy --bucket bucket-base-name--azid--x-s3 --key TARGET_KEY_NAME
 --copy-source SOURCE_BUCKET_NAME/SOURCE_KEY_NAME --part-number 1 --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

詳細については、「AWS Command Line Interface」の「upload-part-copy」を参照してください。

進行中のマルチパートアップロードの一覧表示

ディレクトリバケットへの進行中のマルチパートアップロードをリスト表示するには、AWS SDK または AWS CLI を使用できます。

AWS SDK の使用

SDK for Java 2.x

次の例は、SDK for Java 2.x を使用して、進行中の (不完全な) マルチパートアップロードをリスト表示する方法を示しています。

Example

```
public static void listMultiPartUploads( S3Client s3, String bucketName) {
       try {
           ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
               .bucket(bucketName)
               .build();

           ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
           List MultipartUpload uploads = response.uploads();
           for (MultipartUpload upload: uploads) {
               System.out.println("Upload in progress: Key = \"" + upload.key() +
"\", id = " + upload.uploadId());
           }
     }
     catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
     }
 }
```

SDK for Python

次の例は、SDK for Python を使用して、進行中の (不完全な) マルチパートアップロードをリスト表示する方法を示しています。

Example

```
import logging
```

```
import boto3
from botocore.exceptions import ClientError

def list_multipart_uploads(s3_client, bucket_name):
    '''
    List any incomplete multipart uploads in a directory bucket in e specified gion

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to check for incomplete multipart uploads
    :return: List of incomplete multipart uploads if there are any, None if not
    '''

    try:
        response = s3_client.list_multipart_uploads(Bucket = bucket_name)
        if 'Uploads' in response.keys():
            return response['Uploads']
        else:
            return None
    except ClientError as e:
        logging.error(e)

if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    s3_client = boto3.client('s3', region_name = region)
    multipart_uploads = list_multipart_uploads(s3_client, bucket_name)
    if multipart_uploads is not None:
        print (f'There are {len(multipart_uploads)} ncomplete multipart uploads for
 {bucket_name}')
    else:
        print (f'There are no incomplete multipart uploads for {bucket_name}')
```

AWS CLI の使用

次の例は、AWS CLI を使用して、進行中の (不完全な) マルチパートアップロードをリスト表示する方法を示しています。このコマンドを使用する際は、*###############*を独自の情報に置き換えます。

```
aws s3api list-multipart-uploads --bucket bucket-base-name--azid--x-s3
```

詳細については、「AWS Command Line Interface」の「list-multipart-uploads」を参照してください。

マルチパートアップロードパートの一覧表示

次の例は、ディレクトリバケットへのマルチパートアップロードの一部をリスト表示する方法を示しています。

AWS SDK の使用

SDK for Java 2.x

次の例は、SDK for Java 2.x を使用して、ディレクトリバケットへのマルチパートアップロードの一部をリスト表示する方法を示しています。

```
public static void listMultiPartUploadsParts( S3Client s3, String bucketName, String
 objKey, String uploadID) {

        try {
            ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                .bucket(bucketName)
                .uploadId(uploadID)
                .key(objKey)
                .build();

            ListPartsResponse response = s3.listParts(listPartsRequest);
            ListPart parts = response.parts();
            for (Part part: parts) {
                System.out.println("Upload in progress: Part number = \"" +
 part.partNumber() + "\", etag = " + part.eTag());
            }

        }

        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }


    }
```

SDK for Python

次の例は、SDK for Python を使用して、ディレクトリバケットへのマルチパートアップロードの一部をリスト表示する方法を示しています。

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_parts(s3_client, bucket_name, key_name, upload_id):
    '''
    Lists the parts that have been uploaded for a specific multipart upload to a
 directory bucket.

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that multipart uploads parts have been uploaded to
    :param key_name: Name of the object that has parts uploaded
    :param upload_id: Multipart upload ID that the parts are associated with
    :return: List of parts associated with the specified multipart upload, None if
 there are no parts
    '''
    parts_list = []
    next_part_marker = ''
    continuation_flag = True
    try:
        while continuation_flag:
            if next_part_marker == '':
                response = s3_client.list_parts(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = upload_id
                )
            else:
                response = s3_client.list_parts(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = upload_id,
                    NextPartMarker = next_part_marker
                )
            if 'Parts' in response:
                for part in response['Parts']:
                    parts_list.append(part)
                if response['IsTruncated']:
                    next_part_marker = response['NextPartNumberMarker']
                else:
                    continuation_flag = False
            else:
                continuation_flag = False
```

```
        return parts_list
    except ClientError as e:
        logging.error(e)
        return None

if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'KEY_NAME'
    upload_id = 'UPLOAD_ID'
    s3_client = boto3.client('s3', region_name = region)
    parts_list = list_parts(s3_client, bucket_name, key_name, upload_id)
    if parts_list is not None:
        print (f'{key_name} has {len(parts_list)} parts uploaded to {bucket_name}')
    else:
        print (f'There are no multipart uploads with that upload ID for
 {bucket_name} bucket')
```

AWS CLI の使用

次の例は、AWS CLI を使用して、ディレクトリバケットへのマルチパートアップロードの一部をリスト表示する方法を示しています。このコマンドを使用する際は、*###############*を独自の情報に置き換えます。

```
aws s3api list-parts --bucket bucket-base-name--azid--x-s3 --key KEY_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

詳細については、「AWS Command Line Interface」の「list-parts」を参照してください。

オブジェクトのディレクトリバケットへのコピー

コピーオペレーションを行うと、Amazon S3 内に既に格納されているオブジェクトのコピーが作成されます。ディレクトリバケットと汎用バケット間でオブジェクトをコピーできます。また、単一のバケット内または同じタイプのバケット間でオブジェクトをコピーすることもできます。例えば、ディレクトリバケットからディレクトリバケットへとオブジェクトをコピーできます。

1 回のアトミックオペレーションでコピーできるオブジェクトのサイズは最大 5 GB です。5 GB を超えるオブジェクトをコピーする場合は、マルチパートアップロード API オペレーションを使用する必要があります。詳細については、「ディレクトリバケットでのマルチパートアップロードの使用」を参照してください。

アクセス許可

これを実行するには、次のアクセス許可が必要です。

- あるディレクトリバケットから別のディレクトリバケットにオブジェクトをコピーするには、`s3express:CreateSession` アクセス許可が必要です。
- ディレクトリバケットから汎用バケットにオブジェクトをコピーするには、コピー先バケットにオブジェクトコピーを書き込む `s3express:CreateSession` アクセス許可と `s3:PutObject` アクセス許可が必要です。
- 汎用バケットからディレクトリバケットにオブジェクトをコピーするには、`s3express:CreateSession` アクセス許可とコピーされるソースオブジェクトを読み取る `s3:GetObject` アクセス許可が必要です。

  詳細については、「Amazon Simple Storage Service API リファレンス」の「CopyObject」を参照してください。

暗号化

Amazon S3 は、S3 バケットにコピーされたすべての新しいオブジェクトを自動的に暗号化します。S3 バケットのデフォルトの暗号化設定は常に有効になっており、最低でも Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) に設定されています。

ディレクトリバケットでは、SSE-S3 のみがサポートされています。汎用バケットの場合、SSE-S3 (デフォルト)、AWS Key Management Service (AWS KMS) キーを使用したサーバー側の暗号化 (SSE-KMS)、AWS KMS キーを使用した二層式サーバー側の暗号化 (DSSE-KMS)、またはお客様が用意したキーによるサーバー側の暗号化 (SSE-C) を使用できます。

ディレクトリバケットの SSE-C、SSE-KMS、または DSSE-KMS パラメータをソースまたは送信先として設定するコピーリクエストを実行すると、応答でエラーが返されます。

タグ

ディレクトリバケットは タグをサポートしていません。タグ付きのオブジェクトを汎用バケットからディレクトリバケットにコピーすると、HTTP `501 (Not Implemented)` の応答が返されます。詳細については、「Amazon Simple Storage Service API リファレンス」の「CopyObject」を参照してください。

ETag

S3 Express One Zone のエンティティタグ (ETag) はランダムな英数字文字列であり、MD5 チェックサムではありません。オブジェクトの整合性を確保するには、追加のチェックサムを使用します。

追加のチェックサム

S3 Express One Zone では、アップロードまたはダウンロード中にデータを検証するために使用されるチェックサムアルゴリズムを選択するオプションが提供されます。CRC32、CRC32C、SHA-1、SHA-256 などのセキュアハッシュアルゴリズム (SHA) や巡回冗長検査 (CRC) データ整合性チェックアルゴリズムのいずれかを選択できます。MD5 ベースのチェックサムは S3 Express One Zone ストレージクラスではサポートされていません。

詳細については、「S3 の追加のチェックサムのベストプラクティス」を参照してください。

サポートされている機能

S3 Express One Zone でサポートされている Amazon S3 機能については、「S3 Express One Zone の違いとは」を参照してください。

S3 コンソールの使用 (ディレクトリバケットへのコピー)

汎用バケットまたはディレクトリバケットからディレクトリバケットにオブジェクトをコピーするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. オブジェクトのコピー元のバケットを次のとおり選択します。
   - 汎用バケットからコピーするには、[汎用バケット] タブをクリックします。
   - ディレクトリバケットからコピーするには、[ディレクトリバケット] タブをクリックします。
4. コピーするオブジェクトが含まれる汎用バケットまたはディレクトリバケットを選択します。
5. [オブジェクト] タブを選択します。[オブジェクト] ページで、コピーするオブジェクトの名前の左にあるチェックボックスをオンにします。
6. [Actions (アクション)] メニューで [Copy (コピー)] を選択します。

   [コピー] ページが表示されます。
7. [送信先] で、送信先タイプに [ディレクトリバケット] を選択します。送信先パスを指定するには、[S3 の参照] を選択し、送信先に移動して、送信先の左側にあるオプションボタンを選択します。右下の [Choose destination] (送信先を選択する) を選択します。

または、送信先パスを入力します。

8. [チェックサム] で、既存のチェックサム関数を使用してオブジェクトをコピーするか、既存のチェックサム関数を新しいチェックサム関数に置き換えるかを選択します。オブジェクトをアップロードしたとき、データの整合性を検証するために使用されたチェックサムアルゴリズムを指定するオプションがありました。オブジェクトをコピーするとき、新しい関数を選択するオプションがあります。最初に追加のチェックサムを指定しなかった場合は、[チェックサム] セクションを使用してチェックサムを追加できます。

   > **Note**
   >
   > 同じチェックサム関数を使用することにした場合でも、オブジェクトサイズが 16 MB を超えると、チェックサム値が変更されることがあります。チェックサム値は、マルチパートアップロードのチェックサムの計算方法によって変わる可能性があります。オブジェクトをコピーするときのチェックサムの変更については、「マルチパートアップロードにパートレベルのチェックサムを使用する」を参照してください。

   チェックサム関数を変更するには、[Replace with a new checksum function] (新しいチェックサム関数で置き換える) を選択します。ドロップダウンリストから新しいチェックサム関数を選択します。オブジェクトがコピーされると、指定されたアルゴリズムを使用して新しいチェックサムが計算され、保存されます。

9. 右下の [Copy] (コピー) を選択します。Amazon S3 によってオブジェクトがコピー先にコピーされます。

S3 コンソールの使用 (汎用バケットへのコピー)

ディレクトリバケットから汎用バケットにオブジェクトをコピーするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [ディレクトリバケット] タブをクリックします。
4. コピーするオブジェクトが含まれるディレクトリバケットを選択します。
5. [オブジェクト] タブを選択します。[オブジェクト] ページで、コピーするオブジェクトの名前の左にあるチェックボックスをオンにします。

6. [Actions (アクション)] メニューで [Copy (コピー)] を選択します。

7. [送信先] で、送信先に [汎用バケット] を選択します。送信先パスを指定するには、[S3 の参照] を選択し、送信先に移動して、送信先の左側にあるオプションボタンを選択します。右下の [Choose destination] (送信先を選択する) を選択します。

   または、送信先パスを入力します。

8. [チェックサム] で、既存のチェックサム関数を使用してオブジェクトをコピーするか、既存のチェックサム関数を新しいチェックサム関数に置き換えるかを選択します。オブジェクトをアップロードしたとき、データの整合性を検証するために使用されたチェックサムアルゴリズムを指定するオプションがありました。オブジェクトをコピーするとき、新しい関数を選択するオプションがあります。最初に追加のチェックサムを指定しなかった場合は、[チェックサム] セクションを使用してチェックサムを追加できます。

   > **Note**
   >
   > 同じチェックサム関数を使用することにした場合でも、オブジェクトサイズが 16 MB を超えると、チェックサム値が変更されることがあります。チェックサム値は、マルチパートアップロードのチェックサムの計算方法によって変わる可能性があります。オブジェクトをコピーするときのチェックサムの変更については、「マルチパートアップロードにパートレベルのチェックサムを使用する」を参照してください。

   チェックサム関数を変更するには、[Replace with a new checksum function] (新しいチェックサム関数で置き換える) を選択します。ドロップダウンリストから新しいチェックサム関数を選択します。オブジェクトがコピーされると、指定されたアルゴリズムを使用して新しいチェックサムが計算され、保存されます。

9. 右下の [Copy] (コピー) を選択します。Amazon S3 によってオブジェクトがコピー先にコピーされます。

AWS SDK の使用

SDK for Java 2.x

Example

```
public static void copyBucketObject (S3Client s3, String sourceBucket, String
objectKey, String targetBucket) {
```

```
      CopyObjectRequest copyReq = CopyObjectRequest.builder()
          .sourceBucket(sourceBucket)
          .sourceKey(objectKey)
          .destinationBucket(targetBucket)
          .destinationKey(objectKey)
          .build();
       String temp = "";

       try {
           CopyObjectResponse copyRes = s3.copyObject(copyReq);
           System.out.println("Successfully copied " + objectKey +" from bucket " +
sourceBucket +" into bucket "+targetBucket);
       }

       catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
}
```

AWS CLI の使用

次の copy-object コマンド例は、AWS CLI を使用して、オブジェクトをあるバケットから別のバケットにコピーする方法を示しています。複数のバケットタイプ間でオブジェクトをコピーできます。このコマンドを実行するには、ユーザー入力プレースホルダーをユーザー自身の情報に置き換えます。

```
aws s3api copy-object --copy-source bucket SOURCE_BUCKET/SOURCE_KEY_NAME --
key TARGET_KEY_NAME --bucket TARGET_BUCKET_NAME
```

詳細については、AWS CLI コマンドリファレンスの copy-object を参照してください。

ディレクトリバケットのオブジェクトの削除

Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、または AWS SDK を使用して、Amazon S3 ディレクトリバケットからオブジェクトを削除できます。詳細については、ディレクトリバケットおよびS3 Express One Zone とはを参照してください。

Warning

- オブジェクトを削除すると元に戻せません。
- このアクションは、指定されたすべてのオブジェクトを削除します。フォルダを削除する場合は、削除アクションが完了するのを待ってから、フォルダに新しいオブジェクトを追加します。そうしなければ、新しいオブジェクトも削除される可能性があります。

Note

ディレクトリバケットからプログラムを使用して複数のオブジェクトを削除する場合は、次の点に注意してください。

- `DeleteObjects` リクエストのオブジェクトキーには、空白以外の文字を 1 つ以上含める必要があります。空白文字のみを含めることはできません。
- `DeleteObjects` リクエストのオブジェクトキーには、改行 (\n)、タブ (\t)、キャリッジリターン (\r) 文字を除き、Unicode 制御文字を含めることはできません。

S3 コンソールの使用

オブジェクトを削除するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [ディレクトリバケット] タブをクリックします。
4. 削除するオブジェクトが含まれるディレクトリバケットを選択します。
5. [オブジェクト] タブを選択します。[オブジェクト] リストで、削除するオブジェクト (複数選択可) の左にあるチェックボックスをオンにします。
6. [削除] を選択します。
7. [オブジェクトの削除] ページで、テキストボックスに「**`permanently delete`**」を入力します。
8. [Delete objects] (オブジェクトの削除) を選択します。

AWS SDK の使用

SDK for Java 2.x

Example

次の例では、AWS SDK for Java 2.x を使用してディレクトリバケット内のオブジェクトを削除します。

```
static void deleteObject(S3Client s3Client, String bucketName, String objectKey) {



        try {

            DeleteObjectRequest del = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            s3Client.deleteObject(del);

            System.out.println("Object " + objectKey + " has been deleted");


        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

    }
```

SDK for Python

Example

次の例では、AWS SDK for Python (Boto3) を使用してディレクトリバケット内のオブジェクトを削除します。

```
import logging
import boto3
from botocore.exceptions import ClientError
```

```
def delete_objects(s3_client, bucket_name, objects):
    '''
    Delete a list of objects in a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that contains objects to be deleted; for example,
 'doc-example-bucket--usw2-az1--x-s3'
    :param objects: List of dictionaries that specify the key names to delete
    :return: Response output, else False
    '''

    try:
        response = s3_client.delete_objects(
            Bucket = bucket_name,
            Delete = {
                'Objects': objects
            }
        )
        return response
    except ClientError as e:
        logging.error(e)
        return False


if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    objects = [
        {
            'Key': '0.txt'
        },
        {
            'Key': '1.txt'
        },
        {
            'Key': '2.txt'
        },
        {
            'Key': '3.txt'
        },
        {
            'Key': '4.txt'
        }
    ]
```

```
    s3_client = boto3.client('s3', region_name = region)
    results = delete_objects(s3_client, bucket_name, objects)
    if results is not None:
        if 'Deleted' in results:
            print (f'Deleted {len(results["Deleted"])} objects from {bucket_name}')
        if 'Errors' in results:
            print (f'Failed to delete {len(results["Errors"])} objects from
{bucket_name}')
```

AWS CLI の使用

次の delete-object コマンド例は、AWS CLI を使用してディレクトリバケットからオブジェクトを削除する方法を示しています。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3api delete-object --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
```

詳細については、AWS CLI コマンドリファレンスの delete-object を参照してください。

ディレクトリバケット内のオブジェクトのダウンロード

次のコード例は、GetObject API オペレーションを使用して、Amazon S3 ディレクトリバケット内のオブジェクトからデータを (ダウンロードして) 読み取る方法を示しています。

AWS SDK の使用

SDK for Java 2.x

Example

次のコード例は、AWS SDK for Java 2.x を使用して、ディレクトリバケット内のオブジェクトからデータを読み取る方法を示しています。

```
public static void getObject(S3Client s3Client, String bucketName, String objectKey)
 {
     try {
         GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(objectKey)
            .bucket(bucketName)
```

```
            .build();

         ResponseBytes GetObjectResponse objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
         byte[] data = objectBytes.asByteArray();

         //Print object contents to console
         String s = new String(data, StandardCharsets.UTF_8);
         System.out.println(s);
    }

    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

SDK for Python

Example

次のコード例は、AWS SDK for Python (Boto3) を使用して、ディレクトリバケット内のオブジェクトからデータを読み取る方法を示しています。

```
import boto3
from botocore.exceptions import ClientError
from botocore.response import StreamingBody

def get_object(s3_client: boto3.client, bucket_name: str, key_name: str) ->
 StreamingBody:
    """
    Gets the object.
    :param s3_client:
    :param bucket_name: The bucket that contains the object.
    :param key_name: The key of the object to be downloaded.
    :return: The object data in bytes.
    """
    try:
        response = s3_client.get_object(Bucket=bucket_name, Key=key_name)
        body = response['Body'].read()
        print(f"Got object '{key_name}' from bucket '{bucket_name}'.")
    except ClientError:
        print(f"Couldn't get object '{key_name}' from bucket '{bucket_name}'.")
```

```
        raise
    else:
        return body

def main():
    s3_client = boto3.client('s3')
    resp = get_object(s3_client, 'doc-example-bucket--use1-az4--x-s3', 'sample.txt')
    print(resp)

if __name__ == "__main__":
     main()
```

AWS CLI の使用

次の get-object コマンド例は、AWS CLI を使用して Amazon S3 からオブジェクトをダウンロードする方法を示しています。このコマンドは、ディレクトリバケット *bucket-base-name--azid*--x-s3 からオブジェクト *KEY_NAME* を取得します。オブジェクトは、*LOCAL_FILE_NAME* という名前のファイルにダウンロードされます。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3api get-object --bucket bucket-base-name--azid--x-s3 --
key KEY_NAME LOCAL_FILE_NAME
```

詳細については、AWS CLI コマンドリファレンスの get-object を参照してください。

ディレクトリバケットでの HeadObject の使用

次の AWS SDK と AWS CLI の例は、HeadObject API オペレーションを使用して、オブジェクト自体は返さずに Amazon S3 ディレクトリバケット内のオブジェクトからメタデータを取得する方法を示しています。

AWS SDK の使用

SDK for Java 2.x

Example

```
public static void headObject(S3Client s3Client, String bucketName, String
 objectKey) {
     try {
```

```
        HeadObjectRequest headObjectRequest = HeadObjectRequest
                .builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();
        HeadObjectResponse response = s3Client.headObject(headObjectRequest);
        System.out.format("Amazon S3 object: \"%s\" found in bucket: \"%s\" with
ETag: \"%s\"", objectKey, bucketName, response.eTag());
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
```

AWS CLI の使用

次の head-object コマンド例は、AWS CLI を使用してオブジェクトからメタデータを取得する方法を示しています。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3api head-object --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
```

詳細については、AWS CLI コマンドリファレンスの head-object を参照してください。

S3 Express One Zone のセキュリティ

AWS では、クラウドセキュリティを最優先事項としています。AWS のユーザーは、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを利用できます。セキュリティは、AWS とお客様との間での責任共有です。責任共有モデルでは、この責任がクラウドのセキュリティおよびクラウド内のセキュリティとして説明されています。

- クラウドのセキュリティ — AWS は、AWS クラウド で AWS のサービス を実行するインフラストラクチャを保護する責任を負います。また AWS は、お客様が使用するサービスを安全に提供します。AWS Compliance Programs の一環として、サードパーティーの監査人が定期的にセキュリティの有効性をテストおよび検証します。

  Amazon S3 Express One Zone に適用されるコンプライアンスプログラムについては、「AWS のサービス in Scope by Compliance Program」を参照してください。
- クラウド内のセキュリティ — お客様の責任は、使用する AWS のサービス に応じて異なります。また、お客様は、お客様のデータの機密性、企業の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、S3 Express One Zone を使用する際に責任共有モデルを適用する方法を理解するのに役立ちます。セキュリティとコンプライアンスの目標を満たすように S3 Express One Zone を設定する方法を説明します。また、S3 Express One Zone を使用する際のリソースのモニタリングと保護に役立つその他の AWS のサービス の使用方法についても説明します。

トピック

- データ保護と暗号化
- S3 Express One Zone 向け AWS Identity and Access Management (IAM)
- S3 Express One Zone の IAM アイデンティティベースのポリシー
- S3 Express One Zone のディレクトリバケットポリシーの例
- CreateSession authorization
- S3 Express One Zone のセキュリティのベスト プラクティス

データ保護と暗号化

S3 Express One Zone がデータを暗号化して保護する方法の詳細については、次のトピックを参照してください。

トピック

- Amazon S3 マネージドキーを用いたサーバー側の暗号化 (SSE-S3)
- 転送中の暗号化
- 追加のチェックサム
- データの削除

Amazon S3 マネージドキーを用いたサーバー側の暗号化 (SSE-S3)

デフォルトでは、ディレクトリバケットに保存されているすべてのオブジェクトは、Amazon S3 マネージドキー (SSE-S3) を使用したサーバー側の暗号化を使用して自動的に暗号化されます。ディレクトリバケットへの暗号化されていないアップロードは許可されていません。詳細については、Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)および暗号化によるデータの保護を参照してください。

ディレクトリバケットは、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、AWS Key Management Service (AWS KMS) キーによる二層式サーバー側の暗号化

(DSSE-C)、またはお客様が提供する暗号化キーによるサーバー側の暗号化 (SSE-C) はサポートされません。

転送中の暗号化

S3 Express One Zone には、HTTPS (TLS) 経由でのみアクセスできます。

S3 Express One Zone はリージョン API エンドポイントとゾーン API エンドポイントを使用します。使用する Amazon S3 API オペレーションに応じて、リージョンゾーンエンドポイントまたはゾーンエンドポイントのいずれかが必要です。ゲートウェイ仮想プライベートクラウド (VPC) エンドポイントを通じて、ゾーン API オペレーションとリージョン API オペレーションにアクセスできます。ゲートウェイエンドポイントは追加料金なしで使用できます。リージョン API エンドポイントとゾーン API エンドポイントの詳細については、「S3 Express One Zone のネットワーク」を参照してください。

追加のチェックサム

S3 Express One Zone では、アップロードまたはダウンロード中にデータを検証するために使用されるチェックサムアルゴリズムを選択するオプションが提供されます。CRC32、CRC32C、SHA-1、SHA-256 などのセキュアハッシュアルゴリズム (SHA) や巡回冗長検査 (CRC) データ整合性チェックアルゴリズムのいずれかを選択できます。MD5 ベースのチェックサムは S3 Express One Zone ストレージクラスではサポートされていません。

詳細については、「S3 の追加のチェックサムのベストプラクティス」を参照してください。

データの削除

Amazon S3 コンソール、AWS SDK、AWS Command Line Interface (AWS CLI)、または Amazon S3 REST API を使用して、S3 Express One Zone から単一または複数のオブジェクトを直接削除できます。ディレクトリバケット内のすべてのオブジェクトにはストレージ コストが発生するため、不要になったオブジェクトを削除することをお勧めします。

ディレクトリバケットに保存されているオブジェクトを削除すると、削除されるオブジェクト以外のオブジェクトが親ディレクトリに含まれていない場合、親ディレクトリも再帰的に削除されます。

> **Note**
>
> S3 Express One Zone では、多要素認証 (MFA) 削除と S3 バージョニングはサポートされていません。

S3 Express One Zone 向け AWS Identity and Access Management (IAM)

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に管理するうえで役立つ AWS のサービス です。IAM 管理者は、どのユーザーが認証 (サインイン) して、どのユーザーにリソースの使用を許可する (アクセス許可を持たせる) かを制御します。IAMは追加料金なしでご利用いただけます。

デフォルトでは、ユーザーにはディレクトリバケットと S3 Express One Zone オペレーションのためのアクセス許可はありません。ディレクトリバケットへのアクセス権限を付与するには、IAM を使用してユーザー、グループ、またはロールを作成し、それらのアイデンティティにアクセス許可をアタッチします。IAM の詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

アクセスを提供するには、ユーザー、グループ、またはロールにアクセス許可を追加します。

- AWS IAM Identity Center 内のユーザーとグループ — アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」の手順に従ってください。
- アイデンティティプロバイダーを介して IAM で管理されているユーザー — ID フェデレーションのロールを作成します。詳細については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールの作成」を参照してください。
- IAM ロールとユーザー — ユーザーが引き受けることができるロールを作成します。詳細については、「IAM ユーザーガイド」の「 IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。

デフォルトでは、ディレクトリバケットはプライベートであり、アクセスを明示的に許可されているユーザーのみがアクセスできます。ディレクトリバケットのアクセスコントロール境界は、バケットレベルでのみ設定されます。対照的に、汎用バケットのアクセス制御境界は、バケット、プレフィックス、またはオブジェクトタグレベルで設定できます。この違いは、S3 Express One Zone アクセスのバケットポリシーまたは IAM アイデンティティポリシーに含めることができるリソースがディレクトリバケットのみであるためです。

S3 Express One Zone では、IAM 認証に加えて、`CreateSession` API オペレーションが処理する新しいセッションベースのメカニズムを使用してリクエストを認証および承認します。`CreateSession` を使用して、バケットへの低レイテンシーアクセスを実現する一時的な認証情報をリクエストできます。このような一時的な認証情報は、特定のディレクトリバケットに限定されます。

`CreateSession` を使用するには、最新バージョンの AWS SDK を使用するか AWS Command Line Interface (AWS CLI) を使用することをお勧めします。サポートされている AWS SDK と AWS CLI が、ユーザーに代わってセッションの確立、更新、終了の処理を行います。

セッショントークンはゾーン (オブジェクトレベル) オペレーション (`CopyObject` と `HeadBucket` を除く) でのみ使用し、認証関連のレイテンシーをセッション内の多数のリクエストに分散させます。リージョンのエンドポイント API オペレーション (バケットレベルのオペレーション) には IAM 認証を使用しますが、セッションの管理は不要です。詳細については、S3 Express One Zone 向け AWS Identity and Access Management (IAM)および`CreateSession` authorizationを参照してください。

S3 Express One Zone 向けの IAM の詳細については、次のトピックを参照してください。

トピック

- プリンシパル
- リソース
- S3 Express One Zone 向けアクション
- S3 Express One Zone の条件キー
- API オペレーションの承認と認証方法

プリンシパル

バケットへのアクセスを許可するリソースベースのポリシーを作成する場合は、`Principal` 要素を使用して、そのリソースに対するアクションまたはオペレーションをリクエストできるユーザーまたはアプリケーションを指定する必要があります。ディレクトリバケットポリシーでは、次のプリンシパルを使用できます。

- AWS アカウント。
- IAM ユーザー
- IAM ロール
- フェデレーションユーザー

詳細については、IAM ユーザーガイドPrincipalの を参照してください。

リソース

ディレクトリバケットの Amazon リソースネーム (ARN) には、s3express 名前空間、AWS リージョン、AWS アカウント ID、アベイラビリティーゾーン ID を含むディレクトリバケット名が含まれます。ディレクトリバケットにアクセスしてアクションを実行するには、次の ARN 形式を使用する必要があります。

```
arn:aws:s3express:region:account-id:bucket/base-bucket-name--azid--x-s3
```

ARN の詳細については、「IAM ユーザーガイド」の「Amazon Resource Names (ARNs)」を参照してください。リソースの詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素: Resource」を参照してください。

S3 Express One Zone 向けアクション

IAM ポリシーまたはリソースベースのポリシーで、どの S3 on Outposts アクションを許可または拒否するかを定義します。S3 Express One Zone のアクションは特定の API オペレーションに対応しています。S3 Express One Zone には、Amazon S3 の標準名前空間とは異なる固有の IAM 名前空間があります。この名前空間は s3express です。

s3express:CreateSession アクセス許可を付与すると、CreateSession API オペレーションは ゾーンのエンドポイント API (またはオブジェクトレベル) オペレーションにアクセスする際にセッショントークンを取得できるようになります。このようなセッショントークンは、その他すべての ゾーンのエンドポイント API オペレーションへのアクセスを許可するために使用される認証情報を返します。この結果、IAM ポリシーを使用してゾーンの API オペレーションにアクセス許可を付与する必要がなくなります。代わりに、セッショントークンによりアクセスが実現します。

ゾーンとリージョンのエンドポイント API オペレーションの詳細については、「S3 Express One Zone のネットワーク」を参照してください。CreateSession API オペレーションの詳細については、「Amazon Simple Storage Service API リファレンス」の「CreateSession」を参照してください。.

IAM ポリシーステートメントの Action エレメントでは、以下のアクションを指定できます。ポリシーを使用して、AWS でオペレーションを実行するアクセス許可を付与します。ポリシーでアクションを使用する場合は、通常、同じ名前の API オペレーションまたは CLI コマンドへのアクセスを許可または拒否します。ただし、単独のアクションが複数のオペレーションへのアクセスを制御する場合もあります。バケットレベルのアクションへのアクセスは IAM アイデンティティベースのポリシー (ユーザーまたはロール) でのみ付与でき、バケットポリシーでは付与できません。

S3 Express One Zone のアクションと条件キー

アクション	API	説明	アクセスレベル	条件キー
s3express:CreateBucket	CreateBucket	上がらしいバケット作成のための権限	書き込み	s3express:authType s3express:LocationName s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256
s3express:CreateSession	CreateSession	セッショントークンを作成する権限を付与します。このセッショントークンは、PutObject 、GetObject などのすべてのゾーン (オブジェクトレベル) API オペレーションへのアクセスを許可するために使用されます。	書き込み	s3express:authType s3express:SessionMode s3express:ResourceAccount

アクション	API	説明	アクセスレベル	条件キー
				s3express:signatureversion s3express:signatureAge s3express:TlsVersion s3express:x-amz-content-sha256

アクション	API	説明	アクセスレベル	条件キー
s3express:DeleteBucket	DeleteBucket	URI で指定されたバケットを削除するアクセス許可を付与します。	書き込み	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

アクション	API	説明	アクセスレベル	条件キー
s3express:DeleteBucketPolicy	DeleteBucketPolicy	指定されたバケットのポリシーを削除するアクセス許可を付与します。	権限の管理	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

アクション	API	説明	アクセスレベル	条件キー
s3express:GetBucketPolicy	GetBucketPolicy	指定されたバケットのポリシーを戻すアクセス許可を付与します。	読み取り	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

アクション	API	説明	アクセスレベル	条件キー
s3express:ListAllMyDirectoryBuckets	ListDirectoryBuckets	認証されたリクエストの送信者が所有するすべてのディレクトリバケットを一覧表示するアクセス許可を付与します。	リスト	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

アクション	API	説明	アクセスレベル	条件キー
s3express:PutBucketPolicy	PutBucketPolicy	バケットのバケットポリシーを追加または置き換えるアクセス許可を付与します。	権限の管理	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

S3 Express One Zone の条件キー

S3 Express One Zone は、IAM ポリシーの Condition 要素で使用できる次の条件キーを定義します。これらのキーを使用して、ポリシーステートメントが適用される条件をさらに絞り込むことができます。

条件キー	説明	[Type] (タイプ)
s3express:authType	認証方式でアクセスをフィルタリングします 受信するリクエストが特定の認証方法を使用するように制限するには、このオプションの条件キーを使用します。例えば、この条件キー	文字列

条件キー	説明	[Type] (タイプ)
	を使用すると、リクエスト認証のために HTTP Authorization ヘッダーのみの使用を指定できます。 有効な値:REST-HEADER 、REST-QUERY-STRING|	
s3express:LocationName	CreateBucket API オペレーションへのアクセスを、usw2-az1 などの特定のアベイラビリティーゾーン ID (AZ ID) でフィルタリングします。 値の例: usw2-az1	文字列
s3express:ResourceAccount	リソース所有者の AWS アカウント ID でアクセスをフィルタリングします 特定の AWS アカウント ID が所有するディレクトリバケットへのユーザー、ロール、またはアプリケーションアクセスを制限するには、aws:ResourceAccount または s3express:ResourceAccount 条件キーを使用できます。この条件キーは、AWS Identity and Access Management (IAM) ID ポリシーまたは仮想プライベートクラウド (VPC) エンドポイントポリシーのいずれかで使用できます。例えば、この条件キーを使用して、VPC 内でユーザーの所有していないバケットにクライアントがアクセスできないように制限できます。 値の例: 111122223333	文字列

条件キー	説明	[Type] (タイプ)
`s3express:SessionMode`	`CreateSession` API オペレーションがリクエストしたアクセス許可でアクセスをフィルタリングします。デフォルトでは、このセッションは `ReadWrite` です。この条件キーを使用して、`ReadOnly` アクセスを制限したり、`ReadWrite` アクセスを明示的に拒否したりできます。詳細については、「Amazon Simple Storage Service API リファレンス」の「S3 Express One Zone のディレクトリバケットポリシーの例」と「CreateSession」を参照してください。 有効な値:`ReadWrite` 、`ReadOnly`|	文字列
`s3express:signatureAge`	リクエスト署名の経過時間 (ミリ秒単位) でアクセスをフィルタリングします この条件は、署名付き URL でのみ機能します。 AWS Signature Version 4 では、署名キーは最大 7 日間有効です。したがって、署名の有効期間も最大 7 日間となります。詳細については、「Amazon Simple Storage Service API リファレンス」の「Introduction to Signing Requests」(リクエスト署名の導入) を参照してください。この条件を使用すると、署名の有効期間をさらに制限できます。 値の例: `600000`	数値

条件キー	説明	[Type] (タイプ)
s3express:signatureversion	認証されたリクエストでサポートされる AWS 署名のバージョンを識別します。認証されたリクエストの場合、S3 Express One Zone は Signature Version 4 をサポートします。 有効な値: "AWS4-HMAC-SHA256" (Signature Version 4 を識別)	文字列
s3express:TlsVersion	クライアントが使用する TLS バージョンでアクセスをフィルタリングします s3:TlsVersion 条件キーを使用して、クライアントが使用する TLS バージョンに基づいてディレクトリバケットへのユーザーまたはアプリケーションのア クセスを制限する IAM ポリシー、仮想プライベートクラウドエンドポイント (VPCE) ポリシー、またはバケットポリシーを作成できます。この条件キーを使用して、最小の TLS バージョンを必要とするポリシーを作成することもできます。 値の例: 1.3	数値

条件キー	説明	[Type] (タイプ)
`s3express:x-amz-content-sha256`	バケット内の未署名のコンテンツでアクセスをフィルタリングします この条件キーを使用すると、バケット内にある署名されていないコンテンツを許可しません。 `Authorization` ヘッダーを使用するリクエストに Signature Version 4 を使用する場合、署名計算に `x-amz-content-sha256` ヘッダーを追加してから、その値をハッシュペイロードに設定します。 バケットポリシーでこの条件キーを使用すると、ペイロードが署名されていないアップロードを拒否できます。例: • リクエスト認証において、`Authorization` ヘッダーを使用するアップロードを拒否し、ペイロードには署名しません。詳細については、「Amazon Simple Storage Service API リファレンス」の「Transferring payload in a single chunk」(ペイロードを 1 つのチャンクで転送する) を参照してください。 • 署名付き URL を使用するアップロードを拒否します。署名済み URL は常に `UNSIGNED_PAYLOAD` を持ちます。詳細については、「Amazon Simple Storage Service API リファレンス」の「Authenticating Requests」(認証リクエスト)「Authentication methods」(認証メソッド) を参照してください。 有効な値: `UNSIGNED-PAYLOAD`	文字列

API オペレーションの承認と認証方法

S3 Express One Zone API オペレーションの承認と認証情報は、次の表のとおりです。この表には、API オペレーションごとに、API オペレーション名、IAM アクション、エンドポイントタイプ (リージョンまたはゾーン)、承認メカニズム (IAM またはセッションベース) が表示されています。この表には、クロスアカウントアクセスがサポートされている個所も示されています。バケットレベルのアクションへのアクセスは、バケットポリシーではなく、IAM アイデンティティベースのポリシー (ユーザーまたはロール) でのみ付与です。

API	[エンドポイントタイプ]	IAM アクション	クロスアカウントアクセス
CreateBucket	リージョン別	s3express:CreateBucket	いいえ
DeleteBucket	リージョン別	s3express:DeleteBucket	いいえ
ListDirectoryBuckets	リージョン別	s3express:ListAllMyDirectoryBuckets	いいえ
PutBucketPolicy	リージョン別	s3express:PutBucketPolicy	いいえ
GetBucketPolicy	リージョン別	s3express:GetBucketPolicy	いいえ
DeleteBucketPolicy	リージョン別	s3express:DeleteBucketPolicy	いいえ
CreateSession	ゾーン別	s3express:CreateSession	はい
CopyObject	ゾーン別	s3express:CreateSession	はい
DeleteObject	ゾーン別	s3express:CreateSession	はい
DeleteObjects	ゾーン別	s3express:CreateSession	はい
HeadObject	ゾーン別	s3express:CreateSession	はい
PutObject	ゾーン別	s3express:CreateSession	はい

API	[エンドポイントタイプ]	IAM アクション	クロスアカウントアクセス
GetObjectAttributes	ゾーン別	s3express:CreateSession	はい
ListObjectsV2	ゾーン別	s3express:CreateSession	はい
HeadBucket	ゾーン別	s3express:CreateSession	はい
CreateMultipartUpload	ゾーン別	s3express:CreateSession	はい
UploadPart	ゾーン別	s3express:CreateSession	はい
UploadPartCopy	ゾーン別	s3express:CreateSession	はい
CompleteMultipartUpload	ゾーン別	s3express:CreateSession	はい
AbortMultipartUpload	ゾーン別	s3express:CreateSession	はい
ListParts	ゾーン別	s3express:CreateSession	はい
ListMultipartUploads	ゾーン別	s3express:CreateSession	はい

S3 Express One Zone の IAM アイデンティティベースのポリシー

ディレクトリバケットを作成したり、Amazon S3 Express One Zone ストレージクラスを使用したりする前に、AWS Identity and Access Management (IAM) ロールまたはユーザーに必要なアクセス許可を付与する必要があります。このポリシー例では、CreateSession API オペレーション (ゾーンエンドポイント [オブジェクトレベル] API オペレーションで使用) とすべてのリージョンエンドポイント (バケットレベル) API オペレーションへのアクセスを許可します。このポリシーでは、すべてのディレクトリバケットでの CreateSession API オペレーションの使用が許可されます。た

だし、リージョンエンドポイント API オペレーションは、指定されたディレクトリバケットでの使用のみが許可されます。このポリシーの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "AllowAccessRegionalEndpointAPIs",
             "Effect": "Allow",
             "Action": [
                 "s3express:DeleteBucket",
                 "s3express:DeleteBucketPolicy",
                 "s3express:CreateBucket",
                 "s3express:PutBucketPolicy",
                 "s3express:GetBucketPolicy",
                 "s3express:ListAllMyDirectoryBuckets"
             ],

             "Resource": "arn:aws:s3express:region:account_id:bucket/bucket-base-
name--azid--x-s3/*"
         },
         {
             "Sid": "AllowCreateSession",
             "Effect": "Allow",
             "Action": "s3express:CreateSession",
             "Resource": "*"
         }
     ]
 }
```

S3 Express One Zone のディレクトリバケットポリシーの例

このセクションには、Amazon S3 Express One Zone ストレージクラスで使用するディレクトリバケットポリシーの例を記載しています。これらのポリシーを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

次のバケット ポリシーの例では、AWS アカウント ID *111122223333* が、指定されたディレクトリバケットのデフォルト `ReadWrite` セッションで `CreateSession` API オペレーションを使用することを許可します。このポリシーは、ゾーンエンドポイント (オブジェクトレベル) の API オペレーションへのアクセスを許可します。

Example – デフォルトの **ReadWrite** セッションでの **CreateSession** 呼び出しを許可するバケットポリシー

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccess",
            "Effect": "Allow",
            "Resource": "arn:aws:s3express:us-west-2:account-id:bucket/bucket-base-name--azid--x-s3",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            },
            "Action": [
                "s3express:CreateSession"
            ]
        }
    ]
}
```

Example – **ReadOnly** セッションでの **CreateSession** 呼び出しを許可するバケットポリシー

次のバケットポリシー例では、AWS アカウント ID *111122223333* が CreateSession API オペレーションを使用することを許可しています。このポリシーは、s3express:SessionMode 条件キーと ReadOnly 値を使用して読み取り専用セッションを設定します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadOnlyAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3express:CreateSession",
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "s3express:SessionMode": "ReadOnly"
                }
            }
        }
    ]
}
```

Example — **CreateSession** 呼び出しのためにクロスアカウントアクセスを許可するバケットポリシー

次のバケットポリシーの例では、AWS アカウント ID *111122223333* が、AWS アカウント ID *444455556666* が所有する指定されたディレクトリ バケットに対して CreateSession API オペレーションを使用することを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccount",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": [
                "s3express:CreateSession"
            ],
            "Resource": "arn:aws:s3express:us-west-2:444455556666:bucket/bucket-base-name--azid--x-s3"

        }
    ]
}
```

CreateSession authorization

Amazon S3 Express One Zone は、AWS Identity and Access Management (AWS IAM) 認証とセッションベースの認証の両方をサポートしています。

- S3 Express One Zone でリージョンエンドポイント API オペレーション (バケットレベル、またはコントロールプレーンのオペレーション) を使用するには、セッション管理を必要としない IAM 認証モデルを使用します。アクセス許可はアクションごとに個別に付与されます。詳細については、「S3 Express One Zone 向け AWS Identity and Access Management (IAM)」を参照してください。
- ゾーンエンドポイント API オペレーション (オブジェクトレベルまたはデータプレーンオペレーション) を使用するには、CreateSession API オペレーションを使用して、データリクエストの低レイテンシー認証に最適化されたセッションを作成および管理します。ッションントークンを取得して使用するには、アイデンティティベースのポリシーまたはバケットポリシーでディレクトリバケットの s3express:CreateSession アクションを許可する必要があります。詳細については、「S3 Express One Zone 向け AWS Identity and Access Management (IAM)」を参照してください。Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、または AWS SDK を使用して S3 Express One ゾーンにアクセスしている場合、S3 Express One Zone はユーザーに代わってセッションを作成します。

Amazon S3 REST API を使用する場合、CreateSession API オペレーションを使用して、アクセスキー ID、シークレットアクセスキー、セッショントークン、有効期限を含む一時的なセキュリティ認証情報を取得できます。一時的な認証情報は、IAM ユーザーの認証情報など、長期的なセキュリティ認証情報を使用するのと同じアクセス許可を提供します。ただし、一時的なセキュリティ認証情報にはセッショントークンが含まれている必要があります。

セッションモード

セッションモードはセッションの範囲を定義します。バケットポリシーでは、s3express:SessionMode 条件キーを指定して、ReadWrite セッションや ReadOnly セッションを作成できるユーザーを制御できます。ReadWrite と ReadOnly の詳細については、「Amazon S3 API リファレンス」の「CreateSession の x-amz-create-session-mode パラメータ」を参照してください。バケットポリシー作成の詳細については、「S3 Express One Zone のディレクトリバケットポリシーの例」を参照してください。

セッショントークン

一時的なセキュリティ認証情報を使用して呼び出しを行う場合、呼び出しにはセッショントークンを含める必要があります。セッショントークンは一時的な認証情報とともに返されます。セッショントークンの範囲はディレクトリバケットに限定され、セキュリティ認証情報が有効で有効期限が切れていないことを確認するために使用されます。セッションを保護するため、一時的なセキュリティ認証情報は 5 分後に期限切れになります。

`CopyObject` および `HeadBucket`

一時的なセキュリティ認証情報の範囲は特定のディレクトリバケットに限定され、特定のディレクトリバケットに対するすべてのゾーン (オブジェクトレベル) オペレーション API 呼び出しで自動的に有効になります。その他のゾーンエンドポイント API 操作とは異なり、`CopyObject` と `HeadBucket` は `CreateSession` 認証を使用しません。すべての `CopyObject` リクエストと `HeadBucket` リクエストは IAM 認証情報を使用して認証され、署名される必要があります。ただし、その他のゾーンエンドポイント API オペレーションと同様に `CopyObject` と `HeadBucket` の `s3express:CreateSession` による承認が引き続き行われます。

詳細については、「Amazon Simple Storage Service API リファレンス」の「CreateSession」を参照してください。

S3 Express One Zone のセキュリティのベスト プラクティス

Amazon S3 Express One Zone は、独自のセキュリティポリシーを開発して実装する際に考慮すべきセキュリティ機能を多数提供しています。以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションを提供するものではありません。これらのベストプラクティスは顧客の環境に必ずしも適切または十分でない可能性があるので、処方箋ではなく、あくまで有用な推奨事項とお考えください。

デフォルトのブロックパブリックアクセスとオブジェクト所有権の設定

S3 Express One Zone ストレージクラスを使用するには、S3 ディレクトリバケットを使用する必要があります。ディレクトリバケットは S3 ブロックパブリックアクセスと S3 オブジェクト所有権をサポートします。このような S3 機能は、バケットとオブジェクトへのアクセスを監査して管理するために使用されます。

デフォルトでは、ディレクトリバケットのすべてのブロックパブリックアクセス設定が有効になっています。さらに、オブジェクト所有者はバケット所有者の強制に設定されています。これは、アクセスコントロールリスト (ACL) が無効になっていることを意味します。上記の設定は変更できません。上記の機能の詳細については、「the section called “パブリックアクセスのブロック”」と「the section called “オブジェクト所有者の管理”」を参照してください。

> **Note**
>
> ディレクトリバケットに保存されているオブジェクトへのアクセスは付与できません。アクセスは、ディレクトリバケットにたいしてのみ付与できます。S3 Express One Zone の認証

モデルは、Amazon S3 の認証モデルとは異なります。詳細については、「CreateSession authorization」を参照してください。

認証と認可

S3 Express One Zone の認証と承認メカニズムは、ゾーンエンドポイント API オペレーションまたはリージョンエンドポイント API オペレーションのいずれに対してリクエストを行っているかに応じて異なります。ゾーン API オペレーションはオブジェクトレベル (データプレーン) のオペレーションです。リージョン API オペレーションはバケットレベル (コントロールプレーン) のオペレーションです。

S3 Express One Zone では、レイテンシーを最小限に抑えるように最適化された新しいセッションベースのメカニズム を使用して、ゾーンエンドポイント API オペレーションへのリクエストを認証および承認します。セッションベースの認証では、AWS SDK は CreateSession API オペレーション を使用して一時的な認証情報をリクエストします。これにより、ディレクトリバケットに低レイテンシーでアクセスできます。これらの一時認証情報は特定のディレクトリバケットに限定され、5 分後に期限切れになります。このような一時的な認証情報を使用して、ゾーン (オブジェクトレベル) API 呼び出しに署名できます。詳細については、「CreateSession authorization」を参照してください。

S3 Express One Zone の認証情報を使用したリクエストの署名

S3 Express One Zone 認証情報を使用して、サービス名として s3express を使用して、AWS Signature Version 4 でゾーンエンドポイント (オブジェクトレベル) API リクエストに署名します。リクエストに署名する際は、CreateSession から返されたシークレットキーを使用し、セッショントークンを x-amzn-s3session-token header に渡します。詳細については、「CreateSession」を参照してください。

S3 Express One Zone クラスで サポートされている AWS SDK は、ユーザーに代わって認証情報と署名を管理します。S3 Express One Zone のための AWS SDK を使用して認証情報を更新し、リクエストに署名することをお勧めします。

IAM 認証情報を使用したリクエストの署名

リージョン (バケットレベル) の API 呼び出しはすべて、一時的なセッション認証情報ではなく AWS Identity and Access Management (IAM) 認証情報で認証および署名される必要があります。IAM 認証情報は IAM アイデンティティのアクセスキー ID とシークレットアクセスキーで構成されます。すべ

ての CopyObject リクエストと HeadBucket リクエストは IAM 認証情報を使用して認証され、署名される必要があります。

ゾーン (オブジェクトレベル) オペレーション呼び出しのレイテンシーを最小限に抑えるには、CopyObject と HeadBucket へのリクエストを除き、CreateSession 呼び出しから取得した S3 Express One Zone 認証情報を使用してリクエストに署名することをお勧めします。

AWS CloudTrail を使用します。

AWS CloudTrail は、Amazon S3 のユーザー、ロール、または AWS のサービス が実行したアクションの記録を提供します。CloudTrail によって収集されたデータを使用して、以下の情報を判断できます。

- Amazon S3 に対して行われたリクエスト
- リクエストが行われた IP アドレス
- リクエストを行ったユーザー
- リクエストが行われた時間
- リクエストに関するその他の詳細

AWS アカウントをセットアップすると、CloudTrail はデフォルトで有効になっています。次のリージョンエンドポイント API オペレーション (バケットレベル、またはコントロールプレーンの API オペレーション) が CloudTrail に記録されます。

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- PutBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets

CloudTrail コンソールで最近のイベントを確認できます。Amazon S3 バケットのアクティビティとイベントの継続的なレコードを作成するには、CloudTrail コンソールで証跡を作成できます。詳細については、AWS CloudTrail ユーザーガイドの Creating a trailを参照してください。

> **Note**
>
> S3 Express One Zone の場合、ゾーンエンドポイント (オブジェクトレベルまたはデータプレーン) API オペレーション (`PutObject` または `GetObject` など) の CloudTrail ログ記録はサポートされていません。

AWS モニタリングツールを使用したモニタリングを実装する

モニタリングは、Amazon S3 および AWS ソリューションの信頼性、セキュリティ、可用性、パフォーマンスを維持する上で重要なエレメントです。AWS では、Amazon S3 およびその他の AWS のサービス をモニタリングするのに役立つツールとサービスを提供しています。例えば、Amazon S3 の Amazon CloudWatch メトリクス、特に `BucketSizeBytes` と `NumberOfObjects` ストレージ メトリクスをモニタリングできます。

S3 Express One Zone ストレージ クラスに保存されたオブジェクトは、Amazon S3 の `BucketSizeBytes` と `NumberOfObjects` のストレージメトリクスには反映されません。ただし、S3 Express One Zone では、`BucketSizeBytes` と `NumberOfObjects` のストレージメトリクスがサポートされています。選択したメトリックスを表示するには、`StorageType` ディメンションを指定することで Amazon S3 ストレージクラスと S3 Express One Zone ストレージクラスを区別できます。詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。

詳細については、Amazon CloudWatch によるメトリクスのモニタリングおよびAmazon S3 のモニタリングを参照してください。

Amazon S3 Express One Zone のパフォーマンスの最適化

Amazon S3 Express One Zone は、最もレイテンシーに敏感なアプリケーションに一貫した 1 桁ミリ秒のデータアクセスを提供することを目的として設計された、高パフォーマンスの単一アベイラビリティーゾーン (AZ) S3 ストレージクラスです。S3 Express One Zone は、単一のアベイラビリティーゾーン内で、Amazon Elastic Compute Cloud、Amazon Elastic Kubernetes Service、Amazon Elastic Container Service などの高パフォーマンスオブジェクトストレージと AWS コンピューティングリソースを同じロケーションに配置するオプションを提供する最初の S3 ストレージクラスです。ストレージとコンピューティングリソースを同じロケーションに配置すると、コンピューティングパフォーマンスとコストが最適化され、データ処理速度が向上します。

S3 Express One Zone は、その他の S3 ストレージ クラスと同様のパフォーマンスの伸縮性を提供します。ただし、最初のバイトの読み取りと書き込みのリクエストのレイテンシは一貫して 1 桁ミ

リ秒で、S3 Standard よりも最大 10 倍高速です。S3 Express One Zone は、非常に高い集約レベルまでのバーストスループットをサポートするようにゼロから設計されています。S3 Express One Zone ストレージ クラスは、カスタム構築されたアーキテクチャを使用してパフォーマンスを最適化し、高パフォーマンスハードウェアにデータを保存することで一貫して低いリクエストレイテンシーを実現します。S3 Express One Zone のオブジェクトプロトコルは強化され、認証とメタデータのオーバーヘッドが合理化されています。

アクセス速度をさらに向上し、1 秒あたり数十万のリクエストをサポートするために、S3 Express One Zone は新しいバケットタイプである Amazon S3 ディレクトリバケットにデータを保存します。各 S3 ディレクトリバケットは、1 秒あたり数十万のトランザクション (TPS) をサポートできます。

1 桁ミリ秒のデータアクセス速度と、1 秒あたりの大量のトランザクションに対応できるディレクトリバケットを実現する高パフォーマンスの専用ハードウェアとソフトウェアの組み合わせにより、S3 Express One Zone はリクエスト集中型のオペレーションに最適な Amazon S3 ストレージクラスとなっています。

後続のトピックでは、S3 Express One Zone ストレージクラスを使用するアプリケーションのパフォーマンスを最適化するためのベストプラクティスのガイドラインと設計パターンについて説明します。

トピック

S3 Express One Zone のパフォーマンスガイドラインと設計パターン

Amazon S3 Express One Zone からオブジェクトをアップロードしたり取得したりするアプリケーションを構築する場合は、パフォーマンスを最適化するためのベストプラクティスガイドラインに従ってください。S3 Express One Zone ストレージクラスを使用するには、S3 ディレクトリバケットを作成する必要があります。S3 Express One Zone ストレージ クラスは、S3 汎用バケットでの使用はサポートされていません。

その他すべての Amazon S3 ストレージクラスと S3 汎用バケットのパフォーマンスガイドラインについては、「設計パターンのベストプラクティス: Amazon S3 のパフォーマンスの最適化」を参照してください。

S3 Express One Zone ストレージクラスとディレクトリバケットを使用する際にアプリケーションで最高のパフォーマンスを実現するには、次のガイドラインと設計パターンをお勧めします。

トピック

- S3 Express One Zone ストレージと AWS コンピューティングリソースを同じロケーションに配置する
- ディレクトリバケット
- ディレクトリバケットの水平方向のスケーリング、リクエストの並列処理
- セッションベースの認証を使用する
- S3 の追加のチェックサムのベストプラクティス
- 最新バージョンの AWS SDK と一般的なランタイムライブラリを使用する
- パフォーマンスのトラブルシューティング

S3 Express One Zone ストレージと AWS コンピューティングリソースを同じロケーションに配置する

各ディレクトリのバケットは、バケットの作成時に選択した単一のアベイラビリティーゾーンに保存されます。まず、コンピュートワークロードまたはリソースにローカルなアベイラビリティーゾーンに新しいディレクトリバケットを作成します。これにより、レイテンシーが非常に低い読み取りと書き込みの利用を直ちに開始できます。ディレクトリバケットは、コンピューティングとストレージの間のレイテンシーを低減するために AWS リージョン 内のアベイラビリティーゾーンを選択できる最初の S3 バケットです。

複数のアベイラビリティーゾーンにわたってディレクトリバケットにアクセスすると、レイテンシーが増加します。パフォーマンスを最適化するには、可能な場合は、同じアベイラビリティーゾーンにある Amazon Elastic Container Service、Amazon Elastic Kubernetes Service、Amazon Elastic Compute Cloud のインスタンスからディレクトリバケットにアクセスすることをお勧めします。

ディレクトリバケット

各ディレクトリバケットは、1 秒あたり数十万のトランザクション (TPS) をサポートできます。汎用バケットとは異なり、ディレクトリバケットはキーをプレフィックスではなくディレクトリに階層的に整理します。プレフィックスは、オブジェクトキー名の先頭にある文字列です。プレフィックスは、ディレクトリと同様の方法でデータを整理する方法と考えることができます。ただし、プレフィックスはディレクトリではありません。

プレフィックスは、汎用バケット内のフラットな名前空間にデータを編成します。汎用バケット内のプレフィックスの数には制限はありません。各プレフィックスは 1 秒あたり少なくとも 3,500

PUT/POST/DELETE または 5,500 GET/HEAD リクエストを処理できます。複数のプレフィックス間でリクエストを並列処理してパフォーマンスをスケールすることもできます。読み取りオペレーションと書き込みオペレーションの両方の場合、このスケーリングは瞬時にではなく段階的に行われます。汎用バケットが新しいより高いリクエスト率に合わせてスケールされる間、HTTP ステータスコード 503 (Service Unavailable) エラーが発生する場合があります。

階層型名前空間では、オブジェクトキーの区切り文字が重要です。ただし、サポートされている唯一の区切り文字はスラッシュ (/) です。ディレクトリは区切り文字の境界で定義されます。例えば、`dir1/dir2/file1.txt` オブジェクトキーを使用すると、ディレクトリ `dir1/` と `dir2/` が自動的に作成され、`file1.txt` オブジェクトが `dir1/dir2/file1.txt` パス内の `/dir2` ディレクトリに追加されます。

オブジェクトがディレクトリバケットにアップロードされる際に作成されるディレクトリには、プレフィックスごとの TPS 制限はなく、HTTP 503 (Service Unavailable) エラーの可能性を低減するために自動的に事前にスケールされます。この自動スケーリングにより、アプリケーションは必要に応じてディレクトリ内およびディレクトリ間の読み取り/書き込みリクエストを並列処理できます。

ディレクトリバケットの水平方向のスケーリング、リクエストの並列処理

ディレクトリバケットに複数のリクエストを同時に発行し、リクエストを別々の接続に分散してアクセス可能な帯域幅を最大化することで、最高のパフォーマンスを実現できます。S3 Express One Zone には、バケットへの接続数に制限はありません。同じディレクトリへの同時書き込みが多数発生した場合、個々のディレクトリのパフォーマンスを水平方向かつ自動的にスケールできます。

オブジェクトキーが最初に作成され、そのキー名にディレクトリが含まれると、そのオブジェクトのためにディレクトリが自動的に作成されます。同じディレクトリにそれ以降にオブジェクトをアップロードする場合、ディレクトリを作成する必要がないため、既存のディレクトリにオブジェクトをアップロードする際のレイテンシーが軽減されます。

ディレクトリバケット内のオブジェクトの保存では、浅いディレクトリ構造と深いディレクトリ構造の両方がサポートされています。ただし、ディレクトリバケットは自動的に水平方向にスケールされ、同じディレクトリまたは並行処理するディレクトリの兄弟への同時アップロードのレイテンシーは低下します。

セッションベースの認証を使用する

S3 Express One Zone とディレクトリバケットは、ディレクトリバケットへのリクエストを認証して承認するための新しいセッションベースの認証メカニズムをサポートしています。セッションベースの認証では、AWS SDK は自動的に `CreateSession` API オペレーションを使用して、ディレク

トリバケットへのデータリクエストの低レイテンシー認証に使用できる一時セッショントークンを作成します。

AWS SDK は CreateSession API オペレーションを使用して一時的な認証情報をリクエストし、5 分ごとにユーザーに代わって自動的にトークンを作成して更新します。S3 Express One Zone ストレージクラスのパフォーマンス上の利点を活用するには、AWS SDK を使用して CreateSession API リクエストを開始および管理することをお勧めします。このセッションベースのモデルの詳細については、「CreateSession authorization」を参照してください。

S3 の追加のチェックサムのベストプラクティス

S3 Express One Zone では、アップロードまたはダウンロード中にデータを検証するために使用されるチェックサムアルゴリズムを選択するオプションが提供されます。CRC32、CRC32C、SHA-1、SHA-256 などのセキュアハッシュアルゴリズム (SHA) や巡回冗長検査 (CRC) データ整合性チェックアルゴリズムのいずれかを選択できます。MD5 ベースのチェックサムは S3 Express One Zone ストレージクラスではサポートされていません。

CRC32 は、S3 Express One Zone との間でデータを送信する際に AWS SDK が使用するデフォルトのチェックサムです。S3 Express One Zone ストレージクラスで最高のパフォーマンスを実現するには、CRC32 と CRC32C を使用することをお勧めします。

最新バージョンの AWS SDK と一般的なランタイムライブラリを使用する

一部の AWS SDK では、S3 クライアントのパフォーマンスをさらに向上させる AWS Common Runtime (CRT) ライブラリも提供しています。このような SDK には AWS SDK for Java 2.x、AWS SDK for C++、AWS SDK for Python (Boto3) があります。CRT ベースの S3 クライアントは、S3 Express One Zone との間でオブジェクトを転送します。マルチパートアップロード API オペレーションのバイト範囲フェッチを自動的に使用して水平方向のスケーリング接続を自動化することにより、パフォーマンスと信頼性が向上しています。

S3 Express One Zone ストレージクラスで最高レベルのパフォーマンスを実現するには、CRT ライブラリを含む最新バージョンの AWS SDK を使用するか、AWS Command Line Interface (AWS CLI) を使用することをお勧めします。

パフォーマンスのトラブルシューティング

レイテンシーの影響を受けやすいアプリケーションのリクエストを再試行する

S3 Express One Zone は、追加の調整を必要とせずに一貫したレベルのパフォーマンスを実現できるように設計されています。ただし、タイムアウト値とリトライを積極的に設定すると、レイテン

シーとパフォーマンスを一定に維持できます。AWS SDK には、特定のアプリケーションの許容値に調整できる設定可能なタイムアウト値と再試行値があります。

AWS Common Runtime (CRT) ライブラリと Amazon EC2 インスタンスタイプの組み合わせ

多数の読み取りおよび書き込みオペレーションを実行するアプリケーションは、そうでないアプリケーションに比べて、より多くのメモリやコンピューティング容量を必要とする可能性があります。コンピューティングを多用するワークロードのために Amazon EC2 インスタンスを起動する際は、アプリケーションが必要とする量のリソースを持つインスタンスタイプを選択します。S3 Express One Zone の高パフォーマンスストレージは、システムメモリ容量が大きく、高パフォーマンスストレージを活用できるより強力な CPU や GPU を備えた、より大規模で新しいインスタンスタイプと組み合わせるのが理想的です。また、CRT が有効になっている AWS SDK の最新バージョンを使用することをお勧めします。これにより、読み取りリクエストと書き込みリクエストの並列処理をより高速化できます。

AWS SDK では HTTP REST API の代わりにセッションベースの認証を使用します。

Amazon S3 では、AWS SDK と同じベストプラクティスに従うことで、HTTP REST API リクエストを使用する際のパフォーマンスを最適化することもできます。ただし、S3 Express One Zone で使用されているセッションベースの承認および認証メカニズムの場合、管理には AWS SDK `CreateSession` とそのマネージドセッショントークンを使用することを強くお勧めします。AWS SDK は `CreateSession` API オペレーションを使用して、ユーザーに代わってトークンの作成と更新を自動的に行います。`CreateSession` を使用すると、各リクエストを承認するための AWS Identity and Access Management (IAM) へのリクエストごとの往復のレイテンシーを回避できます。

S3 Express One Zone を使用した開発

Amazon S3 Express One Zone は、オブジェクトストレージをコンピュートリソースと同じ場所に配置するオプションを備えた単一のアベイラビリティーゾーンを選択できる最初の S3 ストレージクラスです。これにより、最高レベルのアクセス速度を実現できます。S3 Express One Zone ストレージクラスでは、S3 ディレクトリバケットを使用してデータを保存します。各ディレクトリバケットは、S3 Express One Zone ストレージクラスを使用して、バケットの作成時に選択できる単一のアベイラビリティゾーンにオブジェクトを保存します。

ディレクトリバケット作成後、レイテンシーが非常に低い読み取りと書き込みの利用を直ちに開始できます。仮想プライベートクラウド (VPC) を介したエンドポイント接続を使用して、ディレクトリバケットと通信したり、ゾーンとリージョン API オペレーションを使用してオブジェクトとディレクトリバケットを管理したりできます。AWS SDK、Amazon S3 コンソール、AWS Command Line

Interface (AWS CLI)、Amazon S3 REST API を使用して S3 Express One Zone ストレージクラスを使用することもできます。

Amazon S3 Express One Zone ストレージクラスは、単一のアベイラビリティーゾーン内で 99.95% の可用性を実現するように設計されており、Amazon S3 サービスレベル契約に基づいています。S3 Express One Zone では、データは単一のアベイラビリティーゾーン内で冗長的に複数のデバイスに保存されます。S3 Express One Zone は、冗長性の損失を迅速に検出して修復することによって、同時デバイス障害を処理するように設計されています。既存のデバイスに障害が発生した場合、S3 Express One Zone はアベイラビリティーゾーン内の新しいデバイスに自動的にリクエストを移します。この冗長性により、アベイラビリティーゾーン内のデータへのアクセスの中断が回避されます。

トピック

- S3 Express One Zone のアベイラビリティーゾーンとリージョン
- リージョンエンドポイントとゾーンエンドポイント
- S3 Express One Zone API オペレーション

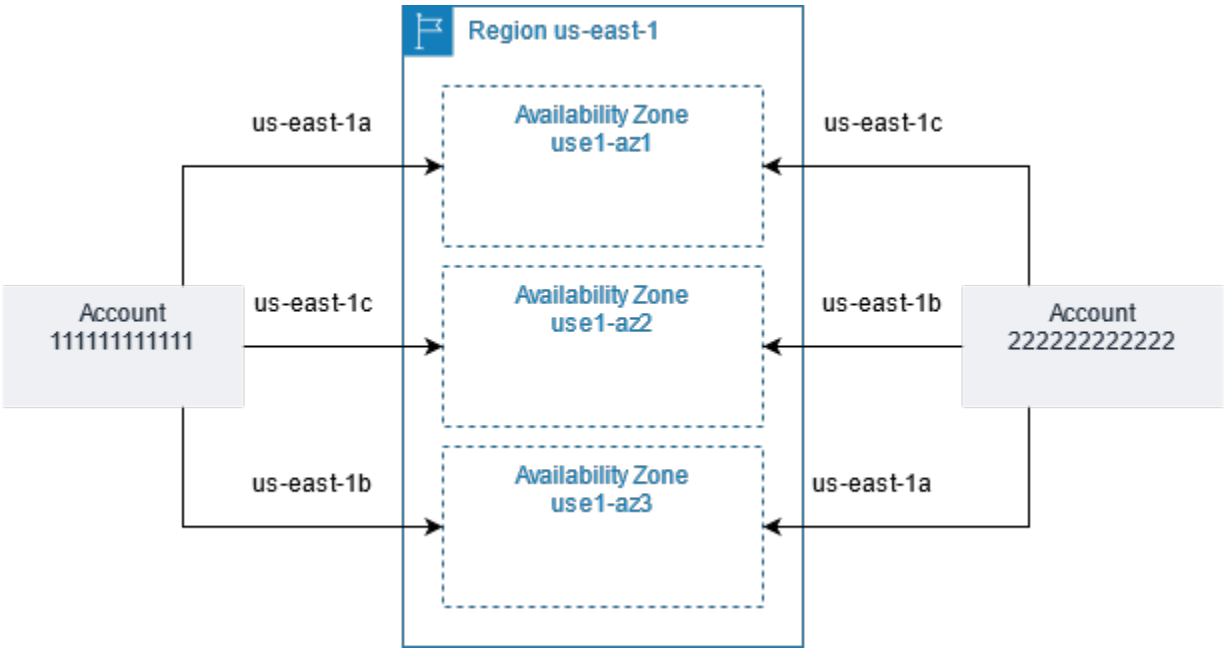
S3 Express One Zone のアベイラビリティーゾーンとリージョン

アベイラビリティーゾーンは、AWS リージョン の冗長電源、ネットワーク、および接続を備えた 1 つ以上の個別のデータセンターです。低レイテンシーでの取り出しを最適化するために、Amazon S3 Express One Zone ストレージクラスのオブジェクトは、コンピューティングワークロードにローカルにある単一のアベイラビリティーゾーンの S3 ディレクトリバケットに冗長的に保存されます。ディレクトリバケットを作成する際は、バケットの配置場所であるアベイラビリティーゾーンと AWS リージョン を選択します。

AWS は、物理的なアベイラビリティーゾーンを AWS アカウント ごとのアベイラビリティーゾーン名にランダムにマップします。この方法は、各リージョンの最初のアベイラビリティーゾーンにリソースが集中するのではなく、AWS リージョン 内のアベイラビリティーゾーン全体にリソースを分散するのに役立ちます。その結果、AWS アカウント のアベイラビリティーゾーン `us-east-1a` は、別の AWS アカウント の `us-east-1a` と同じ物理的な場所を示さない可能性があります。詳細については、Amazon EC2 ユーザーガイドの「リージョンとアベイラビリティゾーン」を参照してください。

アカウント間でアベイラビリティーゾーンを調整するには、アベイラビリティーゾーンの一意で一貫性のある識別子である AZ ID を使用する必要があります。例えば、`use1-az1` は、`us-east-1` リージョンの AZ ID で、すべての AWS アカウント で同じ物理的な場所になります。次の図は、アベイ

ラビリティーゾーン名がアカウントごとに異なるようにマップされている場合でも、AZ ID がすべてのアカウントで同じであるかどうかを説明しています。



S3 Express One Zone では、データは単一のアベイラビリティーゾーン内で冗長的に複数のデバイスに保存されます。S3 Express One Zone は、単一のアベイラビリティーゾーン内で 99.95% の可用性を実現するように設計されており、Amazon S3 サービスレベル契約に基づいています。詳細については、「単一のアベイラビリティーゾーン」を参照してください。

S3 Express One Zone は、次のリージョンとアベイラビリティーゾーンでサポートされています。

S3 Express One Zone をサポートするリージョンとアベイラビリティーゾーン

リージョン名	リージョンコード	アベイラビリティーゾーン ID
米国東部 (バージニア北部)	us-east-1	use1-az4
		use1-az5
		use1-az6
米国西部 (オレゴン)	us-west-2	usw2-az1
		usw2-az3

リージョン名	リージョンコード	アベイラビリティーゾーン ID
		usw2-az4
アジアパシフィック (東京)	ap-northeast-1	apne1-az1
		apne1-az4
欧州 (ストックホルム)	eu-north-1	eun1-az1
		eun1-az2
		eun1-az3

リージョンエンドポイントとゾーンエンドポイント

Virtual Private Cloud (VPC) から Amazon S3 Express One Zone のリージョンエンドポイントとゾーンエンドポイントにアクセスするには、ゲートウェイ VPC エンドポイントを使用できます。ゲートウェイ エンドポイントを作成した後、VPC から S3 Express One Zone を送信先とするトラフィックのルートテーブルにターゲットとして追加できます。ゲートウェイエンドポイントは追加料金なしで使用できます。ゲートウェイ VPC エンドポイントの設定方法の詳細については、「S3 Express One Zone のネットワーク」を参照してください。

S3 Express One Zone を使用している場合、バケットレベル (コントロールプレーン) の API オペレーションはリージョンのエンドポイントを通じて実行でき、リージョンエンドポイント API オペレーションと呼ばれます。リージョンエンドポイント API オペレーションの例には、`CreateBucket`、`DeleteBucket` があります。

ディレクトリバケットを作成した後、ゾーン (オブジェクトレベル、またはデータプレーンエンドポイント API オペレーション) を使用して、ディレクトリバケット内のオブジェクトをアップロードおよび管理できます。ゾーンエンドポイント API オペレーションは、ゾーンエンドポイントを通じて実行できます。ゾーン API オペレーションの例には、`PutObject`、`CopyObject` があります。

S3 Express One Zone API オペレーション

Amazon S3 Express One Zone ストレージクラスは、リージョン (バケットレベル、またはコントロールプレーン) とゾーン (オブジェクト レベル、またはデータ プレーン) の両方のエンドポイント

API オペレーションをサポートします。詳細については、S3 Express One Zone のネットワーク および エンドポイントとゲートウェイ VPC エンドポイント を参照してください。

リージョンエンドポイント API オペレーション

S3 Express One Zone では、次のリージョンエンドポイント API オペレーションがサポートされています。

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets
- PutBucketPolicy

ゾーンエンドポイント API オペレーション

S3 Express One Zone では、 次のゾーンエンドポイント API オペレーションがサポートされています。

- CreateSession
- CopyObject
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- PutObject
- AbortMultipartUpload
- CompleteMultiPartUpload
- CreateMultipartUpload
- ListMultipartUploads

- ListParts
- UploadPart
- UploadPartCopy

Amazon S3 アクセスポイントを使用したデータアクセスの管理

Amazon S3 アクセスポイントは、S3 にデータを保存するあらゆる AWS サービスやお客様のアプリケーションのデータアクセスを簡素化します。アクセスポイントは、バケットにアタッチされた名前付きのネットワークエンドポイントで、S3 オブジェクトのオペレーション (`GetObject` や `PutObject` など) を実行するために使用できます。各アクセスポイントには、そのアクセスポイントを介したすべてのリクエストに S3 が適用する個別のアクセス許可とネットワークコントロールがあります。各アクセスポイントは、基になるバケットにアタッチされたバケットポリシーと連動して機能するカスタマイズされたアクセスポイントポリシーを適用します。仮想プライベートクラウド (VPC) からのリクエストのみを受け付けるようにアクセスポイントを設定することで、プライベートネットワークへの Amazon S3 データアクセスを制限できます。また、アクセスポイントごとにカスタムのブロックパブリックアクセスを設定することもできます。

> **Note**
>
> - アクセスポイントは、オブジェクトに対するオペレーションの実行にのみ使用できます。アクセスポイントを使用して、バケットの変更や削除など、他の Amazon S3 オペレーションを実行することはできません。アクセスポイントをサポートする S3 オペレーションの詳細なリストについては、「AWS サービスとアクセスポイントの互換性」を参照してください。
> - アクセスポイントは、AWS の一部のサービスと機能で動作します。たとえば、アクセスポイントを介して動作するようにクロスリージョンレプリケーションを設定することはできません。S3 アクセスポイントと互換性のある AWS のサービスの詳細なリストについては、「AWS サービスとアクセスポイントの互換性」を参照してください。

このセクションでは、Amazon S3 アクセスポイントの操作方法について説明します。バケットの操作方法の詳細については、「バケットの概要」を参照してください。オブジェクトの操作方法の詳細については、「Amazon S3 オブジェクトの概要」を参照してください。

トピック

- アクセスポイントを使用するための IAM ポリシーの設定
- アクセスポイントの作成
- アクセスポイントの使用

- アクセスポイントの制約と制限

アクセスポイントを使用するための IAM ポリシーの設定

Amazon S3 アクセスポイントは AWS Identity and Access Management (IAM) リソースポリシーをサポートしています。これにより、リソース、ユーザー、その他の条件別にアクセスポイントの使用を制御できます。アプリケーションやユーザーがアクセスポイントを介してオブジェクトにアクセスできるようにするには、アクセスポイントと基になるバケットの両方でリクエストを許可する必要があります。

> **Important**
>
> S3 アクセスポイントをバケットに追加しても、バケット名や Amazon リソースネーム (ARN) でアクセスしたときのバケットの動作は変わりません。バケットに対する既存のすべてのオペレーションは、以前と同じように動作します。アクセスポイントポリシーに含めた制限は、そのアクセスポイントを介したリクエストにのみ適用されます。

IAM リソースポリシーを使用する際、ポリシーを保存する前に、AWS Identity and Access Management Access Analyzer からのセキュリティ警告、エラー、一般的な警告、および提案を解決してください。IAM Access Analyzer は、IAM ポリシーの文法とベストプラクティスに照らしてポリシーチェックを行います。これらのチェックにより、機能的でセキュリティのベストプラクティスに準拠したポリシーを作成するのに、役立つ結果とレコメンデーションが示されます。

IAM Access Analyzer を使用したポリシーの検証の詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer のポリシーの検証」を参照してください。IAM Access Analyzer によって返される警告、エラー、および提案のリストを表示するには、「IAM Access Analyzer ポリシーチェックリファレンス」を参照してください。

アクセスポイントポリシーの例

以下の例は、アクセスポイントを介したリクエストを制御するための IAM ポリシーの作成方法を示しています。

> **Note**
>
> アクセスポイントポリシーで付与されるアクセス許可は、基になるバケットでも同じアクセスが許可される場合にのみ有効です。このためには以下の 2 つの方法があります。

1. (推奨)「アクセスポイントへのアクセスコントロールの委任」の説明に従って、バケットからアクセスポイントにアクセスコントロールを委任します。
2. アクセスポイントポリシーに含まれているものと同じアクセス許可を、基になるバケットのポリシーに追加します。例 1 のアクセスポイントポリシーの例は、必要なアクセスを許可するように基になるバケットポリシーを変更する方法を示しています。

Example 1 - アクセスポイントポリシーによる付与

以下のアクセスポイントポリシーは、アカウント *123456789012* の IAM ユーザー *Jane* に、アカウント *123456789012* のアクセスポイント *my-access-point* を介して、オブジェクト (プレフィックスが *Jane/* であるオブジェクト) を GET および PUT するアクセス許可を付与します。

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Jane"
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point/
object/Jane/*"
    }]
}
```

Note

アクセスポイントポリシーで *Jane* に対して効果的にアクセスを許可するには、基になるバケットでも *Jane* に対して同じアクセスを許可する必要があります。「アクセスポイントへのアクセスコントロールの委任」で説明しているように、バケットからアクセスポイントにアクセスコントロールを委任できます。または、基になるバケットに以下のポリシーを追加して、Jane に必要なアクセス許可を付与できます。Resource エントリはアクセスポイントとバケットのポリシーでは異なります。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Jane"
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/Jane/*"
    }]
}
```

Example 2 - タグ条件付きのアクセスポイントポリシー

以下のアクセスポイントポリシーは、アカウント *123456789012* の IAM ユーザー *Mateo* に、アカウント *123456789012* のアクセスポイント *my-access-point* を介して、オブジェクト (タグキーが *data*、値が *finance* であるオブジェクト) を GET するアクセス許可を付与します。

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect":"Allow",
        "Principal" : {
            "AWS": "arn:aws:iam::123456789012:user/Mateo"
        },
        "Action":"s3:GetObject",
        "Resource" : "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point/
object/*",
        "Condition" : {
            "StringEquals": {
                "s3:ExistingObjectTag/data": "finance"
            }
        }
    }]
}
```

Example 3 – バケットのリスト取得を許可するアクセスポイントポリシー

以下のアクセスポイントポリシーでは、アカウント *123456789012* の IAM ユーザー Arnav に、アカウント *123456789012* のアクセスポイント *my-access-point* の基になるバケットに含まれるオブジェクトを表示するアクセス許可を付与します。

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Arnav"
        },
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point"
    }]
}
```

Example 4 – サービスコントロールポリシー

次のサービスコントロールポリシーは、すべての新しいアクセスポイントを 仮想プライベートクラウド (VPC) ネットワークオリジンで作成することを要求します。このポリシーを適用すると、組織内のユーザーは、インターネットからアクセス可能なアクセスポイントを新規に作成できなくなります。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Deny",
        "Action": "s3:CreateAccessPoint",
        "Resource": "*",
        "Condition": {
            "StringNotEquals": {
                "s3:AccessPointNetworkOrigin": "VPC"
            }
        }
    }]
}
```

Example 5 – S3 オペレーションを VPC ネットワークオリジンに制限するバケットポリシー

次のバケットポリシーは、すべての S3 オペレーションからバケット *example-s3-bucket* へのアクセスを、VPC をネットワークオリジンとするアクセスポイントに制限します。

> **Important**
>
> この例で示しているようなステートメントを使用する前に、アクセスポイントでサポートされない機能 (クロスリージョンレプリケーションなど) が不要であることを確認してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:BypassGovernanceRetention",
                "s3:DeleteObject",
                "s3:DeleteObjectTagging",
                "s3:DeleteObjectVersion",
                "s3:DeleteObjectVersionTagging",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging",
                "s3:ListMultipartUploadParts",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectLegalHold",
                "s3:PutObjectRetention",
                "s3:PutObjectTagging",
                "s3:PutObjectVersionAcl",
                "s3:PutObjectVersionTagging",
                "s3:RestoreObject"
            ],
            "Resource": "arn:aws:s3:::example-s3-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:AccessPointNetworkOrigin": "VPC"
```

```
                }
            }
        }
    ]
}
```

条件キー

S3 アクセスポイントには、IAM ポリシーでリソースへのアクセスを制御するために使用できる条件キーがあります。以下の条件キーは IAM ポリシーの一部にすぎません。すべてのポリシーの例については、「アクセスポイントポリシーの例」、「the section called “アクセスポイントへのアクセスコントロールの委任”」および「the section called “クロスアカウントアクセスポイントへのアクセス許可の付与”」を参照してください。

s3:DataAccessPointArn

この例は、アクセスポイントの ARN の照合に使用できる文字列です。次の例では、リージョン *us-west-2* の AWS アカウント *123456789012* のすべてのアクセスポイントを照合します。

```
"Condition" : {
    "StringLike": {
        "s3:DataAccessPointArn": "arn:aws:s3:us-west-2:123456789012:accesspoint/*"
    }
}
```

s3:DataAccessPointAccount

この例は、アクセスポイントの所有者のアカウント ID を照合するために使用できる文字列演算子です。次の例では、AWS アカウント *123456789012* が所有するすべてのアクセスポイントを照合します。

```
"Condition" : {
    "StringEquals": {
        "s3:DataAccessPointAccount": "123456789012"
    }
}
```

s3:AccessPointNetworkOrigin

この例は、ネットワークオリジン (Internet または VPC) の照合に使用できる文字列演算子です。次の例では、VPC オリジンを持つアクセスポイントのみを照合します。

```
"Condition" : {
    "StringEquals": {
        "s3:AccessPointNetworkOrigin": "VPC"
    }
}
```

Amazon S3 での条件キーの使用についての詳細は、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、条件キー」を参照してください。

アクセスポイントへのアクセスコントロールの委任

バケットのアクセスコントロールをバケットのアクセスポイントに委任できます。以下のバケットポリシーの例では、バケット所有者のアカウントが所有するすべてのアクセスポイントへのフルアクセスを許可しています。したがって、このバケットへのすべてのアクセスは、そのアクセスポイントにアタッチされているポリシーによってコントロールされます。バケットへの直接アクセスを必要としないすべてのユースケースでは、この方法でバケットを設定することをお勧めします。

Example 6 - アクセスコントロールをアクセスポイントに委任するバケットポリシー

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Bucket owner's account
 ID" }
        }
    }]
}
```

クロスアカウントアクセスポイントへのアクセス許可の付与

別のアカウントが所有するバケットへのアクセスポイントを作成するには、まずバケット名とアカウント所有者 ID を指定してアクセスポイントを作成する必要があります。次に、バケット所有者は、アクセスポイントからのリクエストを許可するようにバケットポリシーを更新する必要があります。

アクセスポイントの作成は、アクセスポイントがバケットの内容へのアクセスを提供しないという点で DNS CNAME の作成と似ています。すべてのバケットアクセスはバケットポリシーによって制御されます。次のバケットポリシーの例では、信頼できる AWS アカウント が所有するアクセスポイントからのバケットで、GET および LIST リクエストを許可します。

Bucket ARN は、バケットの ARN に置き換えます。

Example 7 — 別の AWS アカウント にアクセス許可を委任するバケットポリシー

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : ["s3:GetObject","s3:ListBucket"],
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Access point owner's
 account ID" }
        }
    }]
}
```

アクセスポイントの作成

Amazon S3 には、アクセスポイントを作成および管理するための機能が用意されています。S3 アクセスポイントは、AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して作成できます。

デフォルトでは、リージョンごとに AWS アカウントあたり最大 10,000 個のアクセスポイントを作成できます。1 つのリージョンで 1 つのアカウントに 10,000 個を超えるアクセスポイントが必要な場合は、サービスクォータの引き上げをリクエストできます。サービスクォータと引き上げリクエストの詳細については、「AWS 全般のリファレンス」の「AWS Service Quotas」を参照してください。

> **Note**
>
> アクセスポイント名を公開してアクセスポイントの使用を他のユーザーに許可する場合は、アクセスポイント名に機密情報を含めないようにしてください。アクセスポイント名は、ド

メインネームシステム (DNS) と呼ばれるパブリックアクセス可能なデータベースに公開されます。

Amazon S3 アクセスポイントの命名規則

アクセスポイント名は、次の条件を満たす必要があります。

- 1 つの AWS アカウントおよび 1 つのリージョン内で一意である
- DNS 名前付けの制限に従う
- 数字または小文字で始める
- 3 ~ 50 文字の長さにする
- 名前をハイフン (-) で開始または終了することはできません。
- 下線 (_)、大文字、ピリオド (.) は使用できません。
- サフィックス `-s3alias` で終わることはできません。このサフィックスは、アクセスポイントのエイリアス名用に予約されています。詳細については、「S3 バケットアクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

アクセスポイントを作成するには、以下のトピックを参照してください。

トピック

- アクセスポイントの作成
- Virtual Private Cloud に制限されたアクセスポイントの作成
- アクセスポイントへのパブリックアクセスの管理

アクセスポイントの作成

アクセスポイントは、1 つの Amazon S3 バケットにのみ関連付けられます。AWS アカウント でバケットを使用する場合は、まずバケットを作成する必要があります。バケットの作成の詳細については、「Amazon S3 バケットの作成、設定、操作」を参照してください。

バケット名とバケット所有者のアカウント ID がわかっていれば、別の AWS アカウント のバケットに関連するクロスアカウントアクセスポイントを作成することもできます。ただし、クロスアカウントアクセスポイントを作成しても、バケット所有者からアクセス権が付与されるまで、バケット内のデータへのアクセスは許可されません。バケット所有者は、バケットポリシーを通じて、アクセスポ

イント所有者のアカウント (お客様のアカウント) にバケットへのアクセス権を付与する必要があります。詳細については、「クロスアカウントアクセスポイントへのアクセス許可の付与」を参照してください。

デフォルトでは、リージョンごとに AWS アカウントあたり最大 10,000 個のアクセスポイントを作成できます。1 つのリージョンで 1 つのアカウントに 10,000 個を超えるアクセスポイントが必要な場合は、サービスクォータの引き上げをリクエストできます。サービスクォータと引き上げリクエストの詳細については、「AWS 全般のリファレンス」の「AWS Service Quotas」を参照してください。

以下の例は、AWS CLI および S3 コンソールを使用してアクセスポイントを作成する方法を示しています。REST API を使用してアクセスポイントを作成する方法の詳細については、「Amazon Simple Storage Service API リファレンス」の「CreateAccessPoint」を参照してください。

S3 コンソールの使用

アクセスポイントを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ページ上部にあるナビゲーションバーで、現在表示されている AWS リージョン の名前をクリックします。次に、アクセスポイントを作成するリージョンを選択します。
3. 左のナビゲーションペインで、[アクセスポイント] を選択します。
4. [アクセスポイント] ページで、[アクセスポイントの作成] を選択します。
5. [アクセスポイント名] フィールドに、アクセスポイントの名前を入力します。アクセスポイントの名前付けの詳細については、「Amazon S3 アクセスポイントの命名規則」を参照してください。
6. [バケット名] には、アクセスポイントで使用する S3 バケットを指定します。

   アカウントのバケットを使用するには、[このアカウント内のバケットを選択] を選択し、バケット名を入力または参照します。

   別の AWS アカウント でバケットを使用するには、[別のアカウントのバケットを指定] を選択し、バケットの AWS アカウント ID と名前を入力します。

   > **Note**
   >
   > 別の AWS アカウント でバケットを使用している場合は、バケット所有者がアクセスポイントからのリクエストを許可するようにバケットポリシーを更新する必要がありま

す。バケットポリシーの例については、「クロスアカウントアクセスポイントへのアクセス許可の付与」を参照してください。

7. [ネットワークオリジン] を選択してください。[Virtual private cloud (VPC)] を選択した場合は、アクセスポイントで使用する VPC ID を入力します。

   アクセスポイントのネットワークオリジンの詳細については、「Virtual Private Cloud に制限されたアクセスポイントの作成」を参照してください。

8. [Block Public Access settings for this Access Point] (このアクセスポイントのパブリックアクセス設定をブロック) で、アクセスポイントに適用するパブリックアクセスブロック設定を選択します。デフォルトでは、新しいアクセスポイントに対してすべてのブロックパブリックアクセス設定が有効になります。これらの設定を特に無効にする必要がある場合を除いて、すべての設定を有効にしておくことをお勧めします。

   > **Note**
   >
   > アクセスポイントの作成後は、アクセスポイントパブリックアクセスのブロック設定を変更できません。

   アクセスポイントで Amazon S3 パブリックアクセスブロックを使用することについて詳細は、「アクセスポイントへのパブリックアクセスの管理」を参照してください。

9. (オプション) [Access Point policy - optional] (アクセスポイントポリシー - オプション) で、アクセスポイントポリシーを指定します。ポリシーを保存する前に、セキュリティ警告、エラー、一般的な警告、および提案を解決してください。アクセスポイントポリシーの指定の詳細については、「アクセスポイントポリシーの例」を参照してください。

10. [アクセスポイントを作成] を選択します。

AWS CLI の使用

以下の例では、アカウント *111122223333* のバケット *DOC-EXAMPLE-BUCKET* に「*example-ap*」という名前のアクセスポイントを作成します。アクセスポイントを作成するには、以下を指定するリクエストを Amazon S3 に送信します。

- アクセスポイント名。命名規則の詳細については、「the section called “Amazon S3 アクセスポイントの命名規則”」を参照してください。
- アクセスポイントを関連付けるバケットの名前。

- バケットを所有している AWS アカウント のアカウント ID。

```
aws s3control create-access-point --name example-ap --account-id 111122223333 --
bucket DOC-EXAMPLE-BUCKET
```

別の AWS アカウント でバケットを使用してアクセスポイントを作成する場合は、--bucket-account-id パラメータを含めてください。次のコマンド例では、AWS アカウント *444455556666* にあるバケット *DOC-EXAMPLE-BUCKET2* を使用してAWS アカウント *111122223333* にアクセスポイントを作成します。

```
aws s3control create-access-point --name example-ap --account-id 111122223333 --
bucket DOC-EXAMPLE-BUCKET --bucket-account-id 444455556666
```

Virtual Private Cloud に制限されたアクセスポイントの作成

アクセスポイントを作成するときに、そのアクセスポイントをインターネットからアクセス可能にするか、特定の Virtual Private Cloud (VPC) からのリクエストにのみアクセスを制限するかを指定できます。インターネットからアクセス可能なアクセスポイントは、Internet をネットワークオリジンとすると言います。この種のアクセスポイントは、インターネット上のどこからでも利用できます。ただし、このアクセスポイントや、基となるバケット、関連リソース (リクエストされたオブジェクトなど) に関する他のすべてのアクセス制限に従うことを条件とします。指定された VPC からのみアクセス可能なアクセスポイントは VPC をネットワークオリジンとします。Amazon S3 は、アクセスポイントに対するその VPC からのリクエストを除いて、他のすべてのリクエストを拒否します。

> **Important**
>
> アクセスポイントのネットワークオリジンは、アクセスポイントの作成時にのみ指定できます。アクセスポイントの作成後は、そのネットワークオリジンを変更できません。

アクセスポイントを VPC からのアクセスにのみ制限するには、アクセスポイントを作成するリクエストに VpcConfiguration パラメータを含めます。VpcConfiguration パラメータには、アクセスポイントを使用できるようにする VPC ID を指定します。アクセスポイントを介してリクエストが行われた場合、リクエストは VPC から発信されている必要があり、そうでない場合、Amazon S3 はそれを拒否します。

アクセスポイントのネットワークオリジンを取得するには、AWS CLI、AWS SDK、または REST API を使用します。アクセスポイントに VPC 設定が指定されている場合、そのネットワークオリジンは VPC です。それ以外の場合、アクセスポイントのネットワークオリジンは Internet です。

Example

例: VPC アクセスに制限されたアクセスポイントを作成する

次の例では、アカウント 123456789012 のバケット example-bucket に対して vpc-1a2b3c VPC からのアクセスのみを許可する example-vpc-ap という名前のアクセスポイントを作成します。次に、新しいアクセスポイントのネットワークオリジンが VPC であることを確認します。

AWS CLI

```
aws s3control create-access-point --name example-vpc-ap --account-id 123456789012 --
bucket example-bucket --vpc-configuration VpcId=vpc-1a2b3c
```

```
aws s3control get-access-point --name example-vpc-ap --account-id 123456789012

{
    "Name": "example-vpc-ap",
    "Bucket": "example-bucket",
    "NetworkOrigin": "VPC",
    "VpcConfiguration": {
        "VpcId": "vpc-1a2b3c"
    },
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
}
```

VPC でアクセスポイントを使用するには、VPC エンドポイントのアクセスポリシーを変更する必要があります。VPC エンドポイントは、VPC から Amazon S3 へのトラフィックフローを許可します。これらのエンドポイントには、VPC 内のリソースに対して Amazon S3 とのやり取りを許可する方法を制御するアクセスコントロールポリシーがあります。アクセスポイントを経由した VPC か

ら Amazon S3 へのリクエストが成功するのは、VPC エンドポイントポリシーがアクセスポイントおよび基となるバケットの両方へのアクセスを許可した場合のみです。

> **Note**
>
> VPC 内でのみリソースにアクセスできるようにするには、VPC エンドポイントエンドポイントに対してプライベートホストゾーンを必ず作成してください。プライベートホストゾーンを使うには、VPC 設定を変更して、VPC のネットワーク属性 enableDnsHostnames と enableDnsSupport を true に設定してください。

次のポリシーステートメントの例では、awsexamplebucket1 という名前のバケットと example-vpc-ap という名前のアクセスポイントに対する GetObject の呼び出しを許可するように VPC エンドポイントを設定します。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*",
            "arn:aws:s3:us-west-2:123456789012:accesspoint/example-vpc-ap/object/*"
        ]
    }]
}
```

> **Note**
>
> この例の "Resource" 宣言では、Amazon リソースネーム (ARN) を使用してアクセスポイントを指定します。アクセスポイントの ARN の詳細については、「アクセスポイントの使用」を参照してください。

VPC エンドポイントポリシーの詳細については、VPC ユーザーガイドの「Amazon S3 のエンドポイントポリシーの使用」を参照してください。

アクセスポイントへのパブリックアクセスの管理

Amazon S3 アクセスポイントは、アクセスポイントごとに独立したブロックパブリックアクセス設定をサポートしています。アクセスポイントを作成するときに、そのアクセスポイントに適用するブロックパブリックアクセス設定を指定できます。アクセスポイントを経由するすべてのリクエストについて、Amazon S3 は、そのアクセスポイント、基となるバケット、およびバケット所有者のアカウントに関するパブリックアクセスブロック設定を評価します。これらの設定のいずれかで、リクエストをブロックする必要があることが示されると、Amazon S3 はリクエストを拒否します。

S3 のブロックパブリックアクセス機能の詳細については、「Amazon S3 ストレージへのパブリックアクセスのブロック」を参照してください。

> **Important**
>
> - デフォルトでは、アクセスポイントに対してすべてのブロックパブリックアクセス設定が有効になります。アクセスポイントに適用したくない設定がある場合は、明示的に無効にする必要があります。
> - Amazon S3 は、現在、アクセスポイントの作成後におけるアクセスポイントのブロックパブリックアクセス設定の変更をサポートしていません。

Example

例: カスタムのブロックパブリックアクセス設定を使用してアクセスポイントを作成する

この例では、デフォルトではないブロックパブリックアクセス設定を使用して、アカウント 123456789012 のバケット example-bucket に対して example-ap という名前のアクセスポイントを作成します。次に、新しいアクセスポイントの設定を取得して、ブロックパブリックアクセス設定を確認します。

AWS CLI

```
aws s3control create-access-point --name example-ap --account-id
 123456789012 --bucket example-bucket --public-access-block-configuration
 BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=true,RestrictPublicBuckets=t
```

```
aws s3control get-access-point --name example-ap --account-id 123456789012

{
```

```
    "Name": "example-ap",
    "Bucket": "example-bucket",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": false,
        "IgnorePublicAcls": false,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
}
```

アクセスポイントの使用

アクセスポイントで Amazon S3 バケット内のオブジェクトにアクセスするには、AWS Management Console、AWS CLI、AWS SDK、または S3 REST API を使用できます。

アクセスポイントには Amazon リソースネーム (ARN) があります。アクセスポイントの ARN は、バケットの ARN と似ていますが、明示的に型指定され、アクセスポイントのリージョンとアクセスポイントの所有者の AWS アカウント ID をエンコードします。ARN の詳細については、「AWS 全般のリファレンス」の「Amazon リソースネーム (ARN)」を参照してください。

アクセスポイントの ARN は、`arn:aws:s3:`*`region`*`:`*`account-id`*`:accesspoint/`*`resource`* という形式を使用します。例:

- arn:aws:s3:us-west-2:123456789012:accesspoint/test は、リージョン `test` のアカウント `123456789012` によってアクセスポイントが `us-west-2` という名前で所有されていることを示します。
- arn:aws:s3:us-west-2:123456789012:accesspoint/* は、リージョン `123456789012` のアカウント `us-west-2` のすべてのアクセスポイントを示します。

アクセスポイントを介してアクセスされるオブジェクトの ARN は、`arn:aws:s3:`*`region`*`:`*`account-id`*`:accesspoint/`*`access-point-name`*`/object/`*`resource`* という形式を使用します。例:

- arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/unit-01 は、リージョン `unit-01` のアカウント `test` が所有するアクセスポイント `123456789012` を介してアクセスされるオブジェクト `us-west-2` を示します。

- arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/* は、リージョン `test` のアカウント `123456789012` が所有するアクセスポイント `us-west-2` のすべてのオブジェクトを示します。
- arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/unit-01/finance/* は、リージョン `unit-01/finance/` のアカウント `test` が所有するアクセスポイント `123456789012` でus-west-2 をプレフィックスとするすべてのオブジェクトを示します。

S3 アクセスポイントを介したバケットへのアクセス

S3 アクセスポイントでは、仮想ホスト形式のアドレス指定のみがサポートされます。アクセスポイントを介してバケットをアドレス指定するには、次の形式を使用します。

```
https://AccessPointName-AccountId.s3-accesspoint.region.amazonaws.com
```

> **Note**
>
> - アクセスポイント名にダッシュ (-) 文字が含まれている場合は、URL にダッシュを含めて、アカウント ID の前に別のダッシュを挿入します。例えば、リージョン `us-west-2` で アカウント `123456789012` が所有する `finance-docs` という名前のアクセスポイントを使用する場合、適切な URL は `https://finance-docs-123456789012.s3-accesspoint.us-west-2.amazonaws.com` になります。
> - S3 アクセスポイントは HTTP によるアクセスをサポートせず、HTTPS による安全なアクセスのみをサポートします。

トピック

- アクセスポイントのモニタリングとログ記録
- Amazon S3 コンソールでの Amazon S3 アクセスポイントの使用
- S3 バケットアクセスポイントでのバケット形式のエイリアスの使用
- 互換性のある Amazon S3 オペレーションによるアクセスポイントの使用

仮想プライベートクラウド (VPC)をお持ちの場合は、「VPC エンドポイントと S3 アクセスポイントを使用した Amazon S3 アクセスの管理」を参照してください。

アクセスポイントのモニタリングとログ記録

Amazon S3 は、アクセスポイントを介して行われたリクエストと、アクセスポイントを管理する API に対するリクエスト (CreateAccessPoint や GetAccessPointPolicy など) をログに記録します。使用パターンをモニタリングおよび管理するために、アクセスポイントの Amazon CloudWatch Logs リクエストメトリクスを設定することもできます。

トピック

CloudWatch リクエストメトリクス

アクセスポイントを使用しているアプリケーションのパフォーマンスを理解して向上させるために、Amazon S3 リクエストメトリクスで CloudWatch を使用することができます。リクエストメトリクスは、 Amazon S3 リクエストをモニタリングし、オペレーションの問題をすばやく特定して対応するのに役立ちます。

デフォルトでは、 リクエストメトリクスはバケットレベルで利用可能です。ただし、共有プレフィクス、オブジェクトタグ、またはアクセスポイントを使用して、リクエストメトリクスのフィルタを定義できます。アクセスポイントフィルタを作成すると、リクエストメトリクスの設定に、指定したアクセスポイントへのリクエストが含まれます。メトリクスの受信、アラームの設定、およびダッシュボードへのアクセスにより、このアクセスポイントで実行されたオペレーションをリアルタイムで表示できます。

コンソールで設定するか、Amazon S3 API を使用して、リクエストメトリクスをオプトインする必要があります。リクエストメトリクスは、処理のレイテンシーの後に 1 分間隔で使用できます。リクエストメトリクスは、CloudWatch カスタムメトリクスと同じ料金レートで請求されます。詳細については、「Amazon CloudWatch の料金」を参照してください。

アクセスポイントでフィルタリングするリクエストメトリクスの設定を作成する方法は、プレフィックス、オブジェクトタグ、またはアクセスポイントでのメトリクス設定の作成 を参照してください。

リクエストログ

サーバーアクセスロギングおよび AWS CloudTrail を使用して、アクセスポイントを介して行われたリクエストと、アクセスポイントを管理する API に対するリクエスト (CreateAccessPoint や GetAccessPointPolicy, など) をログに記録します。

アクセスポイントを介して行われたリクエストの CloudTrail ログエントリには、ログの `resources` セクションにアクセスポイント ARN が含まれます。

例えば、次の設定があるとします。

- `my-image.jpg` という名前のオブジェクト を含むリージョン `us-west-2` の `DOC-EXAMPLE-BUCKET1` という名前のバケット
- `DOC-EXAMPLE-BUCKET1` に関連付けられた `my-bucket-ap` という名前のアクセスポイント
- `123456789012` の AWS アカウント ID

以下の例は、`resources`前述の設定の CloudTrail ログエントリの セクションを示しています。

```
"resources": [
        {"type": "AWS::S3::Object",
            "ARN": "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/my-image.jpg"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::Bucket",
            "ARN": "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::AccessPoint",
            "ARN": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-bucket-ap"
        }
    ]
```

S3 サーバーのアクセスログの詳細については、「サーバーアクセスログによるリクエストのログ記録」を参照してください。AWS CloudTrail の詳細については、AWS CloudTrail ユーザーガイドの AWS CloudTrail とは を参照してください。

Amazon S3 コンソールでの Amazon S3 アクセスポイントの使用

このセクションでは、AWS Management Consoleを使用して Amazon S3 アクセスポイントを管理および使用する方法について説明します。開始する前に、次の手順に従って、管理または使用するアクセスポイントの詳細ページに移動します。

トピック

- アカウントのすべてのアクセスポイントの一覧表示
- バケットのアクセスポイントの一覧表示

- アクセスポイントの設定詳細の表示
- アクセスポイントの使用
- アクセスポイントのパブリックアクセスブロック設定の表示
- アクセスポイントポリシーの編集
- アクセスポイントの削除

アカウントのすべてのアクセスポイントの一覧表示

AWS アカウント で作成されたすべてのアクセスポイントを一覧表示する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ページ上部にあるナビゲーションバーで、現在表示されている AWS リージョン の名前をクリックします。次に、アクセスポイントをリスト表示するリージョンを選択します。
3. コンソールの左側のナビゲーションペインで、[access points] (アクセスポイント) を選択します。
4. [アクセスポイント] ページの [アクセスポイント] で、この AWS リージョン のアクセスポイントを確認します。
5. (任意) [リージョン] ドロップダウンメニューの隣にあるテキストフィールドに名前を入力して、アクセスポイントを名前で検索します。
6. 管理または使用するアクセスポイントの名前を選択します。

バケットのアクセスポイントの一覧表示

AWS アカウント にある 1 つのバケットのすべてのアクセスポイントを一覧表示する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ページ上部のナビゲーションバーで、現在表示されている AWS リージョン を選択してから、アクセスポイントをリスト表示するリージョンを選択します。
3. コンソールの左側のナビゲーションペインで、[バケット] を選択します。
4. [バケット] ページで、アクセスポイントを一覧表示するバケットの名前を選択します。
5. バケットの詳細ページで、[access points] (アクセスポイント) タブを選択します。
6. 管理または使用するアクセスポイントの名前を選択します。

アクセスポイントの設定詳細の表示

1. 「アカウントのすべてのアクセスポイントの一覧表示」の説明に従って、詳細を表示するアクセスポイントのアクセスポイント詳細ページに移動します。
2. [access point overview] (アクセスポイントの概要) で、選択したアクセスポイントの設定の詳細とプロパティを表示します。

アクセスポイントの使用

1. 「アカウントのすべてのアクセスポイントの一覧表示」の説明に従って、使用するアクセスポイントのアクセスポイント詳細ページに移動します。
2. [オブジェクト] タブで、アクセスポイント経由でアクセスする 1 つまたは複数のオブジェクトの名前を選択します。オブジェクトオペレーションページで、コンソールに、現在使用しているアクセスポイントを示すラベルがバケット名の下に表示されます。アクセスポイントを使用する際は、アクセスポイントのアクセス許可で許可されているオブジェクト操作のみを実行できます。

> **Note**
>
> - コンソールビューには、バケット内のすべてのオブジェクトが常に表示されます。この手順で説明しているようにアクセスポイントを使用すると、これらのオブジェクトに実行できるオペレーションは制限されますが、オブジェクトがバケット内に存在するかどうかの確認は制限されません。
> - S3 マネジメントコンソールでは、Virtual Private Cloud (VPC) アクセスポイントを使用したバケットリソースへのアクセスはサポートされていません。VPC アクセスポイントからバケットのリソースにアクセスするには、AWS CLI、AWS SDK、または Amazon S3 REST API を使用します。

アクセスポイントのパブリックアクセスブロック設定の表示

1. 「アカウントのすべてのアクセスポイントの一覧表示」の説明に従って、設定を表示するアクセスポイントのアクセスポイント詳細ページに移動します。
2. [Permissions] を選択します。
3. [access point policy] (アクセスポイントポリシー) で、アクセスポイントの [Block Public Access] (パブリックアクセスブロック) 設定を確認します。

> **Note**
>
> アクセスポイントの作成後は、アクセスポイントの [パブリックアクセスブロック] 設定を変更することはできません。

アクセスポイントポリシーの編集

1. 「アカウントのすべてのアクセスポイントの一覧表示」の説明に従って、ポリシーを編集するアクセスポイントのアクセスポイント詳細ページに移動します。
2. [Permissions] を選択します。
3. [access point policy] (アクセスポイントポリシー) で、[Edit] (編集) を選択します。
4. テキストフィールドにアクセスポイントポリシーを入力します。コンソールに、アクセスポイントの Amazon リソースネーム (ARN) が自動的に表示されます。これをポリシーで使用できます。

アクセスポイントの削除

1. 「アカウントのすべてのアクセスポイントの一覧表示」の説明に従って、アカウントまたは特定のバケットのアクセスポイントのリストに移動します 。
2. 削除するアクセスポイントの名前の横にあるオプションボタンを選択します。
3. [削除] を選択します。
4. 表示されるテキストフィールドにアクセスポイントの名前を入力して、アクセスポイントを削除することを確認し、[削除] を選択します。

S3 バケットアクセスポイントでのバケット形式のエイリアスの使用

アクセスポイントを作成すると、Amazon S3 はデータアクセス用の Amazon S3 バケット名の代わりに使用できるエイリアスを自動的に生成します。このアクセスポイントエイリアスは、アクセスポイントのデータプレーンオペレーションにおいて、Amazon リソースネーム (ARN) の代わりに使用することができます。これらのオペレーションのリストについては、「AWS サービスとアクセスポイントの互換性」を参照してください。

以下は、*my-access-point* という名前のアクセスポイントの ARN とアクセスポイントのエイリアスの例です。

- ARN — arn:aws:s3:*region*:*account-id*:accesspoint/*my-access-point*
- アクセスポイントエイリアス — *my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b*-s3alias

ARN の詳細については、「AWS 全般のリファレンス」の「Amazon リソースネーム (ARN)」を参照してください。

アクセスポイントのエイリアス名

アクセスポイントのエイリアス名は、Amazon S3 バケットと同じ名前空間内に作成されます。このエイリアス名は自動的に生成され、変更できません。アクセスポイントのエイリアス名は、有効な Amazon S3 バケット名のすべての要件を満たしており、次の部分で構成されています。

access point prefix-metadata-s3alias

> **Note**
>
> -s3alias サフィックスは、アクセスポイントのエイリアス名用に予約されており、バケット名やアクセスポイント名には使用できません。Amazon S3 バケット命名規則の詳細については、「バケットの名前付け」を参照してください。

アクセスポイントエイリアスの使用例と制限事項

アクセスポイントを採用する場合、大幅なコード変更を必要とせずに、アクセスポイントのエイリアス名を使用できます。

アクセスポイントを作成すると、次の例に示すように、Amazon S3 によってアクセスポイントのエイリアス名が自動的に生成されます。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control create-access-point --bucket example-s3-bucket1 --name my-access-point --
account-id 111122223333
{
    "AccessPointArn":
    "arn:aws:s3:region:111122223333:accesspoint/my-access-point",
    "Alias": "my-access-point-aqfqprnstn7aefdfbarligizwgyfouse1a-s3alias"
}
```

このアクセスポイントのエイリアス名は、あらゆるデータプレーンオペレーションにおいて、Amazon S3 のバケット名の代わりに使用することができます。これらのオペレーションのリストについては、「AWS サービスとアクセスポイントの互換性」を参照してください。

次の get-object コマンドの AWS CLI 例では、バケットのアクセスポイントエイリアスを使用して、指定されたオブジェクトに関する情報を返します。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3api get-object --bucket my-access-point-aqfqprnstn7aefdfbarligizwgyfouse1a-
s3alias --key dir/my_data.rtf my_data.rtf

{
    "AcceptRanges": "bytes",
    "LastModified": "2020-01-08T22:16:28+00:00",
    "ContentLength": 910,
    "ETag": "\"00751974dc146b76404bb7290f8f51bb\"",
    "VersionId": "null",
    "ContentType": "text/rtf",
    "Metadata": {}
}
```

制限事項

- お客様はエイリアスを設定できません。
- アクセスポイントでは、エイリアスの削除、変更、無効化はできません。
- このアクセスポイントのエイリアス名は、一部のデータプレーンオペレーションにおいて、Amazon S3 のバケット名の代わりに使用することができます。これらのオペレーションのリストについては、「S3 オペレーションとアクセスポイントの互換性」を参照してください。
- アクセスポイントのエイリアス名は Amazon S3 コントロールプレーンオペレーションに使用することはできません。Amazon S3 コントロールプレーンオペレーションの一覧は、「Amazon Simple Storage Service API リファレンス」の「Amazon S3 コントロール」を参照してください。
- Amazon S3 コンソールの 移動オペレーションの移動元または移動先として S3 アクセスポイントのエイリアスを使用することはできません。
- エイリアスはAWS Identity and Access Management (IAM) ポリシーでは使用できません。
- エイリアスは S3 サーバーアクセスログのログ記録送信先として使用することはできません。
- エイリアスは AWS CloudTrail ログのログ記録送信先として使用することはできません。

- Amazon SageMaker GroundTruth は、アクセスポイントのエイリアスをサポートしていません。

互換性のある Amazon S3 オペレーションによるアクセスポイントの使用

以下の例は、Amazon S3 の互換性のあるオペレーションでアクセスポイントを使用する方法を示しています。

トピック

AWS サービスとアクセスポイントの互換性

Amazon S3 アクセスポイントのエイリアスを使用すると、S3 バケット名を必要とするすべてのアプリケーションでアクセスポイントを簡単に使用できます。S3 のデータにアクセスするために S3 バケット名を使用する任意の場所で、S3 アクセスポイントのエイリアスを使用できます。詳細については、「アクセスポイントエイリアスの使用例と制限事項」を参照してください。

S3 オペレーションとアクセスポイントの互換性

アクセスポイントを介して、次の Amazon S3 API のサブセットを使用してバケットにアクセスできます。以下に示すすべてのオペレーションで、アクセスポイント ARN またはアクセスポイントエイリアスのいずれかを許可できます。

S3 オペレーション

- `AbortMultipartUpload`
- `CompleteMultipartUpload`
- `CopyObject` (同じリージョンへのコピーのみ)

- CreateMultipartUpload
- DeleteObject
- DeleteObjectTagging
- GetBucketAcl
- GetBucketCors
- GetBucketLocation
- GetBucketNotificationConfiguration
- GetBucketPolicy
- GetObject
- GetObjectAcl
- GetObjectAttributes
- GetObjectLegalHold
- GetObjectRetention
- GetObjectTagging
- HeadBucket
- HeadObject
- ListMultipartUploads
- ListObjects
- ListObjectsV2
- ListObjectVersions
- ListParts
- Presign
- PutObject
- PutObjectLegalHold
- PutObjectRetention
- PutObjectAcl
- PutObjectTagging
- RestoreObject
- UploadPart
- UploadPartCopy (同じリージョンへのコピーのみ)

アクセスポイントを使用したオブジェクトのリクエスト

次の例では、my-image.jpg リージョンのアカウント ID prod が所有するアクセスポイント 123456789012 を使用してオブジェクト us-west-2 をリクエストし、ダウンロードしたファイルを download.jpg として保存します。

AWS CLI

```
aws s3api get-object --key my-image.jpg --bucket arn:aws:s3:us-
west-2:123456789012:accesspoint/prod download.jpg
```

アクセスポイントエイリアスを使用したオブジェクトのアップロード

次の例では、リージョン us-west-2 のアカウント ID 123456789012 が所有するアクセスポイントエイリアス my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b-s3alias を使用してオブジェクト my-image.jpg をアップロードします。

AWS CLI

```
aws s3api put-object --bucket my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b-
s3alias --key my-image.jpg --body my-image.jpg
```

アクセスポイントを使用したオブジェクトの削除

次の例では、my-image.jpg リージョンのアカウント ID prod が所有するアクセスポイント 123456789012 を使用してオブジェクト us-west-2 を削除します。

AWS CLI

```
aws s3api delete-object --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/prod
 --key my-image.jpg
```

アクセスポイントエイリアスを使用したオブジェクトの一覧表示

次の例では、リージョン us-west-2 のアカウント ID 123456789012 が所有するアクセスポイントエイリアス my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b-s3alias を使用してオブジェクトを一覧表示します。

AWS CLI

```
aws s3api list-objects-v2 --bucket my-access-point-
hrzrlukc5m36ft7okagglf3gmwluquse1b-s3alias
```

アクセスポイントを使用したオブジェクトへのタグセットの追加

次の例では、us-west-2 リージョンのアカウント ID 123456789012 が所有するアクセスポイント prod を使用して既存のオブジェクト my-image.jpg にタグセットを追加します。

AWS CLI

```
aws s3api put-object-tagging --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/
prod --key my-image.jpg --tagging TagSet=[{Key="finance",Value="true"}]
```

アクセスポイントを使用した ACL によるアクセス許可の付与

次の例では、us-west-2 リージョンのアカウント ID prod が所有するアクセスポイント my-image.jpg を使用して既存のオブジェクト 123456789012 に ACL を適用します。

AWS CLI

```
aws s3api put-object-acl --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/prod
 --key my-image.jpg --acl private
```

アクセスポイントの制約と制限

Amazon S3 アクセスポイントには以下の制約と制限があります。

- 各アクセスポイントは 1 つのバケットにのみ関連付けられます。このバケットは、アクセスポイントの作成時に指定する必要があります。作成後のアクセスポイントを別のバケットに関連付けることはできません。ただし、アクセスポイントを削除して、別のアクセスポイントを同じ名前で作成し、新しいアクセスポイントを別のバケットに関連付けることはできます。
- アクセスポイント名は特定の条件を満たす必要があります。アクセスポイントの名前付けの詳細については、「Amazon S3 アクセスポイントの命名規則」を参照してください。
- アクセスポイントを作成した後、その Virtual Private Cloud (VPC) 設定を変更することはできません。

- アクセスポイントのポリシーのサイズは 20 KB に制限されています。
- リージョンごとに AWS アカウント あたり最大 10,000 個のアクセスポイントを作成できます。1 つのリージョンで 1 つのアカウントに 10,000 個を超えるアクセスポイントが必要な場合は、サービスクォータの引き上げをリクエストできます。サービスクォータと引き上げリクエストの詳細については、「AWS 全般のリファレンス」の「AWS Service Quotas」を参照してください。
- 1,000 個以上のアクセスポイントを所有している AWS リージョン では、Amazon S3 コンソールでは、アクセスポイントを名前で検索することはできません。
- アクセスポイントを S3 レプリケーションのレプリケーション先として使用することはできません。レプリケーションの詳細については、「オブジェクトのレプリケーション」を参照してください。
- Amazon S3 コンソールの 移動オペレーションの移動元または移動先として S3 アクセスポイントのエイリアスを使用することはできません。
- アクセスポイントにアドレス指定できるのは、仮想ホスト形式の URL だけです。仮想ホスティング形式のアドレス指定の詳細については、「Amazon S3 バケットに対するアクセスと一覧表示」を参照してください。
- アクセスポイントの機能を制御する API オペレーション (`PutAccessPoint` や `GetAccessPointPolicy` など) は、クロスアカウントコールをサポートしていません。
- REST API を使用してアクセスポイントへのリクエストを行う場合は、AWS Signature Version 4 を使用する必要があります。リクエストの認証の詳細については、Amazon Simple Storage Service API リファレンスのリクエストの承認 (AWS Signature Version 4) を参照してください。
- アクセスポイントは HTTPS 経由のリクエストのみをサポートしています。Amazon S3 は、HTTP 経由で行われたすべてのリクエストに対して HTTP リダイレクトで自動的に応答し、リクエストを HTTPS にアップグレードします。
- アクセスポイントは匿名アクセスをサポートしていません。
- クロスアカウントアクセスポイントは、バケット所有者から権限が付与されるまで、データへのアクセスを許可しません。バケット所有者は常にデータを完全に管理できるため、クロスアカウントアクセスポイントからのリクエストを許可するにはバケットポリシーを更新する必要があります。バケットポリシーの例を表示するには、「アクセスポイントを使用するための IAM ポリシーの設定」を参照してください。
- Amazon S3 コンソールでクロスアカウントアクセスポイントを表示すると、[アクセス] 列に [不明] と表示されます。Amazon S3 コンソールは、関連するバケットとオブジェクトにパブリックアクセスが許可されているかどうかを判断できません。特定のユースケースでパブリック設定が必要でない限り、ユーザーとバケット所有者がアクセスポイントとバケットへのすべてのパブリックア

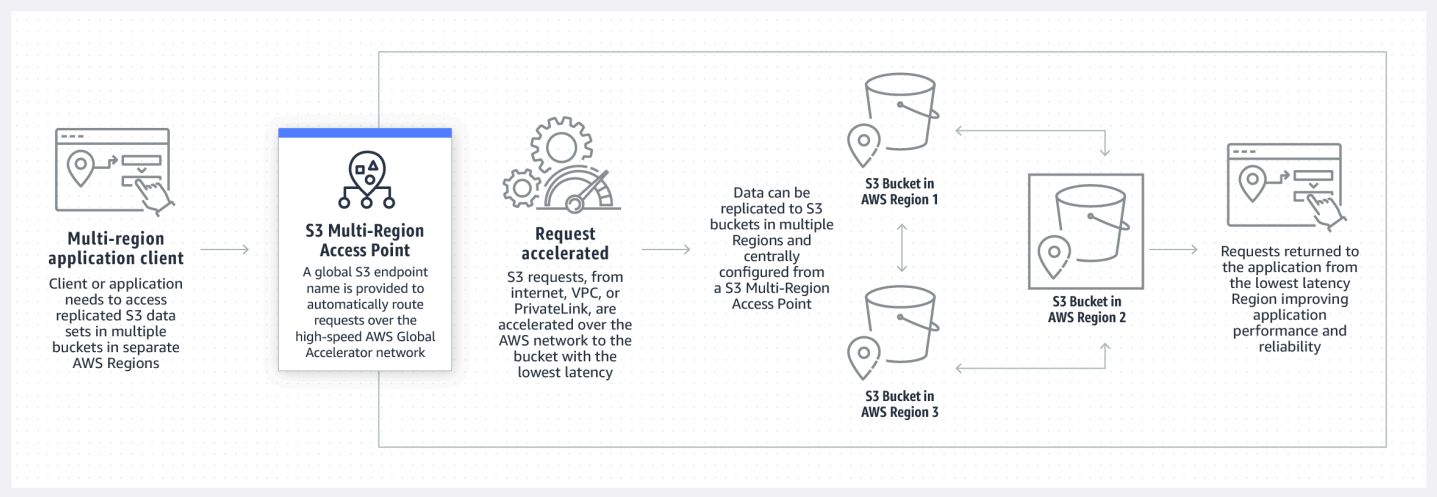
クセスをブロックすることをお勧めします。詳細については、「Amazon S3 ストレージへのパブリックアクセスのブロック」を参照してください。

Amazon S3 マルチリージョンアクセスポイント

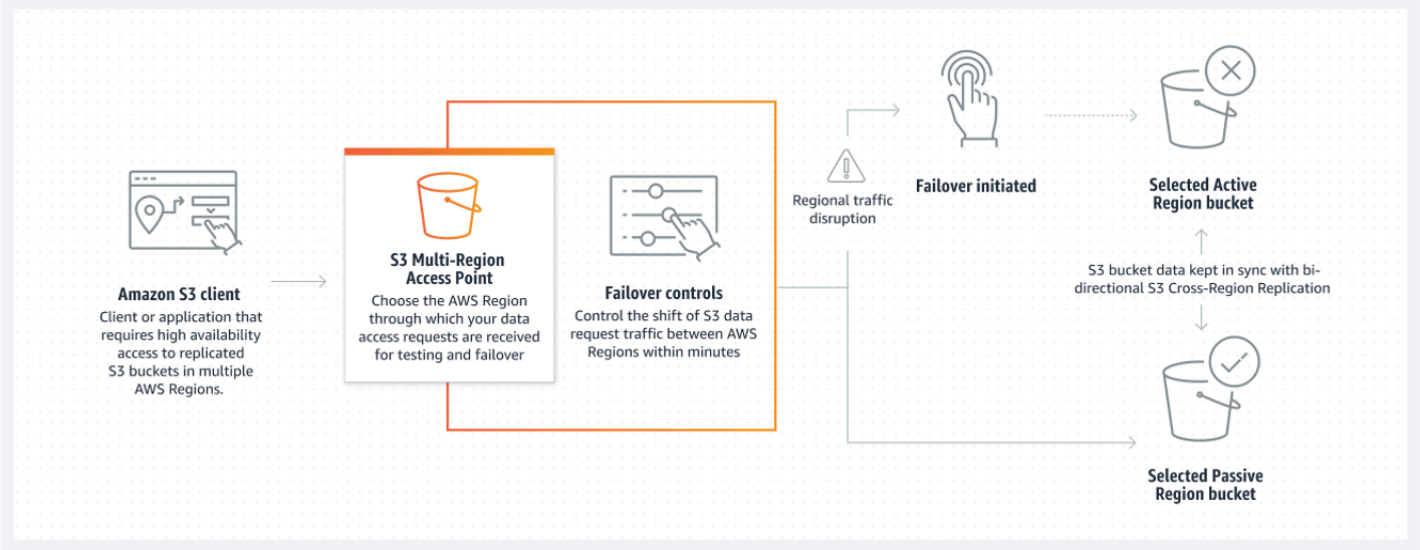
Amazon S3 マルチリージョンアクセスポイントを使用すると、複数の AWS リージョン にある S3 バケットからのリクエストをアプリケーションが実行するために使用できるグローバルエンドポイントを作成できます。マルチリージョンアクセスポイントを使用して、単一のリージョンで使用するのと同じシンプルなアーキテクチャでマルチリージョンアプリケーションを構築し、世界中のどこでもこれらのアプリケーションを実行することができます。マルチリージョンのアクセスポイントは、輻輳したパブリックインターネット経由でリクエストを送信する代わりに、Amazon S3 へのインターネットベースのリクエストを高速化する組み込みのネットワーク耐障害性を実現します。マルチリージョンアクセスポイントのグローバルエンドポイントに対して行われたアプリケーションリクエストは、AWS Global Accelerator を使用して、AWS グローバルネットワークを介して、アクティブなルーティングステータスを持つ最も近い S3 バケットに自動的にルーティングされます。

マルチリージョンアクセスポイントを作成するときは、そのマルチリージョンアクセスポイントを介して提供されるデータを保存する AWS リージョン のセットを指定します。S3 クロスリージョンレプリケーション (CRR) を使用して、それらのリージョンのバケット間でデータを同期します。その後、マルチリージョンアクセスポイントグローバルエンドポイントを介してデータを要求または書き込みできます。Amazon S3 は、最も近い利用可能なリージョンからレプリケートされたデータセットへのリクエストを自動的に処理します。マルチリージョンアクセスポイントは、AWS PrivateLink for Amazon S3 を使用するアプリケーションを含め、Amazon Virtual Private Cloud (VPC) で実行されているアプリケーションとも互換性があります。

次のイメージは、アクティブ-アクティブ構成の Amazon S3 マルチリージョンアクセスポイントを図で示したものです。この図は、Amazon S3 リクエストが最も近いアクティブな AWS リージョンに自動的にルーティングされる様子を示しています。



次のイメージは、アクティブ/パッシブ構成の Amazon S3 マルチリージョンアクセスポイントを図で示したものです。この図は、Amazon S3 データアクセストラフィックを制御してアクティブ-パッシブの AWS リージョン 間でフェイルオーバーする方法を示しています。



マルチリージョンアクセスポイントの使用方法の詳細については、「チュートリアル: Amazon S3 マルチリージョンアクセスポイントの使用の開始方法」を参照してください。

トピック

- マルチリージョンアクセスポイントの作成
- AWS PrivateLink で使用するマルチリージョンアクセスポイントの設定
- マルチリージョンアクセスポイントを使用したリクエスト

マルチリージョンアクセスポイントの作成

Amazon S3 でマルチリージョンアクセスポイントを作成するには、以下の作業を行います。

- マルチリージョンアクセスポイントの名前を指定します。
- マルチリージョンアクセスポイントのリクエストを処理するそれぞれの AWS リージョン でバケットを 1 つ選択します。
- マルチリージョンアクセスポイントに Amazon S3 パブリックアクセスブロックを設定します。

このすべての情報は、Amazon S3 が非同期で処理する作成リクエストで提供します。Amazon S3 は、非同期作成リクエストのステータスをモニタリングするために使用できるトークンを提供しています。

ポリシーを保存する前に、AWS Identity and Access Management Access Analyzer でセキュリティ警告、エラー、一般的な警告、および提案を解決してください。IAM Access Analyzer は、IAM ポリシーの文法およびベストプラクティス に対してポリシーチェックを行います。これらのチェックにより、機能的でセキュリティのベストプラクティスに準拠したポリシーを作成するのに、役立つ結果と実行可能なレコメンデーションが示されます。IAM Access Analyzer を使用したポリシーの検証の詳細については、IAM ユーザーガイドの IAM Access Analyzer のポリシーの検証を参照してください。IAM Access Analyzer によって返される警告、エラー、および提案のリストを表示するには、IAM Access Analyzer ポリシーチェックリファレンスを参照してください。

API を使用する場合、マルチリージョンアクセスポイントの作成リクエストは非同期です。マルチリージョンアクセスポイントを作成するリクエストを送信すると、Amazon S3 はリクエストを同期的に承認します。次に、作成リクエストの進行状況を追跡するために使用できるトークンをすぐに返します。マルチリージョンアクセスポイントを作成および管理するための非同期要リクエストの追跡の詳細については、「マルチリージョンアクセスポイントでのサポートされている API オペレーションの使用」を参照してください。

マルチリージョンアクセスポイントを作成すると、そのアクセスコントロールポリシーを作成できます。それぞれのマルチリージョンアクセスポイントには、ポリシーを関連付けることができます。マルチリージョンアクセスポイントポリシーは、リソース、ユーザー、またはその他の条件別にマルチリージョンアクセスポイントの使用を制限するために使用できるリソースベースのポリシーです。

> **Note**
>
> アプリケーションまたはユーザーがマルチリージョンアクセスポイントを介してオブジェクトにアクセスできるようにするには、次の両方のポリシーでリクエストを許可する必要があります。
>
> - マルチリージョンアクセスポイントのアクセスポリシー
> - オブジェクトを含む基になるバケットのアクセスポリシー
>
> 2 つのポリシーが異なる場合は、より制限の厳しいポリシーが優先されます。
> マルチリージョンアクセスポイントのアクセス許可管理を簡素化するために、バケットからマルチリージョンアクセスポイントにアクセスコントロールを委任できます。詳細について

は、「the section called “マルチリージョンアクセスポイントポリシーの例”」を参照してください。

マルチリージョンアクセスポイントでバケットを使用しても、既存のバケット名または Amazon リソースネーム (ARN) を使用してバケットにアクセスする場合のバケットの動作は変わりません。バケットに対する既存のすべてのオペレーションは、以前と同じように動作します。マルチリージョンアクセスポイントポリシーに含めた制限は、そのマルチリージョンアクセスポイントを介したリクエストにのみ適用されます。

マルチリージョンアクセスポイントのポリシーは、作成後に更新できますが、ポリシーを削除することはできません。ただし、マルチリージョンアクセスポイントポリシーを更新して、すべてのアクセス許可を拒否することができます。

トピック

- Amazon S3 マルチリージョンアクセスポイントの命名規則
- Amazon S3 マルチリージョンアクセスポイントのバケットを選択するためのルール
- Amazon S3 マルチリージョンアクセスポイントを作成する
- Amazon S3 のマルチリージョンアクセスポイントを使用したパブリックアクセスのブロック
- Amazon S3 マルチリージョンアクセスポイントの設定の詳細を表示する
- マルチリージョンアクセスポイントの削除

Amazon S3 マルチリージョンアクセスポイントの命名規則

マルチリージョンアクセスポイントを作成するときは、名前を付けます。これは、選択した文字列です。マルチリージョンアクセスポイントの作成後は名前を変更できません。名前は、AWS アカウント で一意であり、マルチリージョンアクセスポイントの制約および制限事項 に記載されている命名要件に準拠している必要があります。マルチリージョンアクセスポイントを識別しやすくするために、自分にとって意味のある名前、組織にとって意味のある名前、またはシナリオを反映した名前を使用します。

この名前は、`GetMultiRegionAccessPoint` および `PutMultiRegionAccessPointPolicy` などのマルチリージョンアクセスポイント管理オペレーションを呼び出す際に使用します。この名前は、マルチリージョンアクセスポイントへのリクエストの送信には使用されないため、マルチリージョンアクセスポイントを使用してリクエストを行うクライアントに対して公開する必要はありません。

Amazon S3 がマルチリージョンアクセスポイントを作成すると、自動的にエイリアスが割り当てられます。このエイリアスは、.mrap で終わる一意の英数字文字列です。エイリアスは、マルチリージョンアクセスポイントのホスト名と Amazon リソースネーム (ARN) を構築するために使用されます。また、完全修飾名は、マルチリージョンアクセスポイントのエイリアスにも基づいています。

マルチリージョンアクセスポイントの名前をエイリアスから特定することはできません。そのため、イリアスを公開しても、マルチリージョンアクセスポイントの名前、目的、所有者を公開する危険性はありません。Amazon S3 は、新しいマルチリージョンアクセスポイントごとにエイリアスを選択し、エイリアスを変更することはできません。マルチリージョンアクセスポイントのアドレス指定方法の詳細については、「マルチリージョンアクセスポイントを使用したリクエスト」を参照してください。

マルチリージョンアクセスポイントのエイリアスは、時間が経過しても一意であり、マルチリージョンアクセスポイントの名前や構成には基づいていません。マルチリージョンアクセスポイントを作成し、それを削除し、同じ名前と構成で別のものを作成した場合、2 番目のマルチリージョンアクセスポイントのエイリアスは最初のものと異なります。新しいマルチリージョンアクセスポイントは、以前のマルチリージョンアクセスポイントと同じエイリアスを持つことはできません。

Amazon S3 マルチリージョンアクセスポイントのバケットを選択するためのルール

それぞれのマルチリージョンアクセスポイントは、リクエストを処理するリージョンに関連付けられています。マルチリージョンアクセスポイントは、それぞれのリージョンの 1 つのバケットだけに関連付けられる必要があります。マルチリージョンアクセスポイントを作成するために、リクエストで各バケットの名前を指定します。マルチリージョンアクセスポイントをサポートするバケットは、マルチリージョンアクセスポイントを所有している同じ AWS アカウント にあることも、他の AWS アカウント にあることもできます。

1 つのバケットを複数のマルチリージョンアクセスポイントで使用できます。

> **Important**
>
> - マルチリージョンアクセスポイントに関連付けられるバケットは、作成時にのみ指定できます。作成後は、マルチリージョンアクセスポイント設定からバケットを追加、変更、または削除することはできません。バケットを変更するには、マルチリージョンアクセスポイント全体を削除し、新しいリージョンアクセスポイントを作成する必要があります。

- マルチリージョンアクセスポイントの一部であるバケットは削除できません。マルチリージョンアクセスポイントに添付されたバケットを削除する場合は、まずマルチリージョンアクセスポイントを削除します。
- 別のアカウントが所有しているバケットをマルチリージョンアクセスポイントに追加する場合、バケット所有者は、バケットポリシーを更新して、マルチリージョンアクセスポイントにアクセス許可を付与する必要もあります。そうしない場合、マルチリージョンアクセスポイントはそのバケットからデータを取得できません。このようなアクセスを許可する方法を示すポリシーの例については、「マルチリージョンアクセスポイントポリシーの例」を参照してください。
- すべてのリージョンがマルチリージョンアクセスポイントをサポートしているわけではありません。サポートされているリージョンのリストを確認するには、「マルチリージョンアクセスポイントの制約および制限事項」を参照してください。

バケット間でデータを同期するレプリケーションルールを作成できます。これらのルールを使用すると、ソースバケットからコピー先バケットにデータを自動的にコピーできます。バケットをマルチリージョンアクセスポイントに接続しても、レプリケーションの動作には影響しません。マルチリージョンアクセスポイントを使用したレプリケーションの設定については、後のセクションで説明します。

Important

マルチリージョンアクセスポイントにリクエストを行うと、マルチリージョンアクセスポイントは、マルチリージョンアクセスポイント内のバケットのデータコンテンツを認識しません。そのため、リクエストを受け取ったバケットには、リクエストされたデータが含まれていない可能性があります。マルチリージョンアクセスポイントに関連付けられた Amazon S3 バケットに一貫性のあるデータセットを作成するには、S3 クロスリージョンレプリケーション (CRR) を設定することをお勧めします。詳細については、「マルチリージョンアクセスポイントで使用するためのレプリケーションの設定」を参照してください。

Amazon S3 マルチリージョンアクセスポイントを作成する

以下の例では、Amazon S3 コンソールを使用してマルチリージョンアクセスポイントを作成する方法を示しています。

S3 コンソールの使用

マルチリージョンアクセスポイントを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、[Multi-Region Access Points] (マルチリージョンアクセスポイント) を選択します。
3. [マルチリージョンアクセスポイントの作成] を選択して、マルチリージョンアクセスポイントの作成を開始します。
4. [マルチリージョンアクセスポイント] ページの [マルチリージョンアクセスポイント名] フィールドに、マルチリージョンアクセスポイントの名前を入力します。
5. このマルチリージョンアクセスポイントに関連付けるバケットを選択します。自分のアカウントにあるバケットを選択することも、他のアカウントからバケットを選択することもできます。

> **Note**
>
> 自分のアカウントまたは他のアカウントから、少なくとも 1 つのバケットを追加する必要があります。また、マルチリージョンアクセスポイントは、AWS リージョン ごとに 1 つのバケットしかサポートしないことにも注意してください。したがって、同じリージョンから 2 つのバケットを追加することはできません。デフォルトで無効になっている AWS リージョン はサポートされません。

- アカウントにあるバケットを追加するには、[バケットの追加] を選択します。アカウント内のすべてのバケットのリストを表示します。バケットを名前で検索するか、バケット名をアルファベット順に並べ替えることができます。
- 別のアカウントからバケットを追加するには、[他のアカウントからバケットを追加] を選択します。他のアカウントでバケットを検索したり参照したりすることはできないため、正確なバケット名とAWS アカウント ID がわかっていることを確認してください。

> **Note**
>
> 有効なAWS アカウント ID とバケット名を入力する必要があります。また、バケットはサポートされているリージョンにある必要があり、ない場合は、マルチリージョンアクセスポイントを作成しようとすると エラーが発生します。マルチリージョンアク

セスポイントをサポートするリージョンのリストについては、「マルチリージョンアクセスポイントの制約および制限事項」を参照してください。

6. (オプション) 追加したバケットを削除する必要がある場合は、[削除] を選択します。

> **Note**
>
> 作成が完了した後に、このマルチリージョンアクセスポイントのバケットを追加したり削除したりすることはできません。

7. [このマルチリージョンアクセスポイントのパブリックアクセス設定をブロック] で、マルチリージョンアクセスポイントに適用するブロックパブリックアクセス設定を選択します。デフォルトでは、新しいマルチリージョンアクセスポイントに対してすべてのブロックパブリックアクセス設定が有効になります。これらの設定を特に無効にする必要がある場合を除いて、すべての設定を有効にしておくことをお勧めします。

> **Note**
>
> マルチリージョンアクセスポイントを作成した後に、マルチリージョンアクセスポイントのブロックパブリックアクセス設定を変更することはできません。したがって、パブリックアクセスをブロックする場合は、マルチリージョンアクセスポイントを作成する前に、パブリックアクセスがなくてもアプリケーションが正しく動作することを確認してください。

8. [マルチリージョンアクセスポイントの作成] を選択します。

> **Important**
>
> 別のアカウントが所有しているバケットをマルチリージョンアクセスポイントに追加する場合、バケット所有者は、バケットポリシーを更新して、マルチリージョンアクセスポイントにアクセス許可を付与する必要もあります。そうしない場合、マルチリージョンアクセスポイントはそのバケットからデータを取得できません。このようなアクセスを許可する方法を示すポリシーの例については、「マルチリージョンアクセスポイントポリシーの例」を参照してください。

AWS CLI の使用

AWS CLI を使用して、マルチリージョンアクセスポイントを作成できます。マルチリージョンアクセスポイントを作成するときは、それがサポートするすべてのバケットを提供する必要があります。マルチリージョンアクセスポイントを作成した後に、マルチリージョンアクセスポイントにバケットを追加することはできません。

次の例では、AWS CLI を使用して、2 つのバケットを持つマルチリージョンアクセスポイントを作成します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control create-multi-region-access-point --account-id 111122223333 --details '{
        "Name": "simple-multiregionaccesspoint-with-two-regions",
        "PublicAccessBlock": {
            "BlockPublicAcls": true,
            "IgnorePublicAcls": true,
            "BlockPublicPolicy": true,
            "RestrictPublicBuckets": true
        },
        "Regions": [
            { "Bucket": "example-s3-bucket1" },
            { "Bucket": "example-s3-bucket2" }
        ]
    }' --region us-west-2
```

Amazon S3 のマルチリージョンアクセスポイントを使用したパブリックアクセスのブロック

それぞれのマルチリージョンアクセスポイントには、Amazon S3 パブリックアクセスブロックの個別の設定があります。これらの設定は、マルチリージョンアクセスポイントと基となるバケットを所有する AWS アカウント のブロックパブリックアクセス設定と連動して機能します。

Amazon S3 がリクエストを承認すると、これらの設定の最も制限の厳しい組み合わせが適用されます。これらのリソース (マルチリージョンアクセスポイント所有者アカウント、基となるバケット、またはバケット所有者アカウント) のブロックパブリックアクセス設定が、リクエストされたアクションまたはリソースへのアクセスをブロックすると、Amazon S3 はリクエストを拒否します。

これらの設定を特に無効にする必要がある場合を除いては、すべての設定を有効にしておくことをお勧めします。デフォルトでは、マルチリージョンアクセスポイントに対してすべてのブロックパブリックアクセス設定が有効になります。ブロックパブリックアクセスが有効になっている場合、マ

ルチリージョンアクセスポイントはインターネットベースのリクエストを受け付けることができません。

> **Important**
>
> マルチリージョンアクセスポイントを作成した後にブロックパブリックアクセス設定を変更することはできません。

Amazon S3 のブロックパブリックアクセスの詳細については、「Amazon S3 ストレージへのパブリックアクセスのブロック」を参照してください。

Amazon S3 マルチリージョンアクセスポイントの設定の詳細を表示する

以下の例では、Amazon S3 コンソールを使用してマルチリージョンアクセスポイントの設定の詳細を表示する方法を示しています。

S3 コンソールの使用

マルチリージョンアクセスポイントを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、[Multi-Region Access Points] (マルチリージョンアクセスポイント) を選択します。
3. 設定の詳細を表示するマルチリージョンアクセスポイントの名前を選択します。

   - [プロパティ] タブには、マルチリージョンアクセスポイントに関連付けられているすべてのバケット、作成日、Amazon リソースネーム (ARN)、およびエイリアスが表示されます。AWS アカウント ID 列には、マルチリージョンアクセスポイントに関連付けられている外部アカウントが所有するバケットもすべて表示されます。
   - [アクセス許可] タブには、このマルチリージョンアクセスポイントに関連付けられたバケットに適用されるブロックパブリックアクセス設定が表示されます。マルチリージョンアクセスポイントを作成している場合は、マルチリージョンアクセスポイントのポリシーを表示することもできます。[アクセス許可] ページの [情報] アラートには、このマルチリージョンアクセスポイントの [パブリックアクセスはブロックされています] 設定が有効になっているすべてのバケット (自分のアカウントと他のアカウント内) も一覧表示されます。

- [レプリケーションとフェイルオーバー] タブには、マルチリージョンアクセスポイントに関連付けられているバケットと、バケットが置かれているリージョンのマップビューが表示されます。データを取得するアクセス許可のない別のアカウントのバケットがある場合、そのリージョンはレプリケーションの概要マップ上で赤くマークされ、レプリケーションステータスの取得中にエラーが発生した AWS リージョン であることを示します。

  > **Note**
  >
  > 外部アカウントのバケットからレプリケーションステータス情報を取得するには、バケット所有者がバケットポリシーで `s3:GetBucketReplication` アクセス許可を付与する必要があります。

  このタブには、マルチリージョンアクセスポイントで使用されているリージョンのレプリケーションメトリクス、レプリケーションルール、フェイルオーバーステータスも表示されます。

AWS CLI の使用

AWS CLI を使用して、マルチリージョンアクセスポイントの設定の詳細を表示できます。

次の AWS CLI の例では、現在のマルチリージョンアクセスポイント設定を取得します。このコマンドの例を実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control get-multi-region-access-point --account-id 111122223333 --name example-s3-bucket1
```

マルチリージョンアクセスポイントの削除

次の手順では、Amazon S3 コンソールを使用してマルチリージョンアクセスポイントを削除する方法を説明します。

マルチリージョンアクセスポイントを削除しても、マルチリージョンアクセスポイントに関連付けられたバケットは削除されず、マルチリージョンアクセスポイント自体だけが削除されます。

S3 コンソールの使用

マルチリージョンアクセスポイントを削除するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、[Multi-Region Access Points] (マルチリージョンアクセスポイント) を選択します。
3. マルチリージョンアクセスポイントの名前の横にあるオプションボタンを選択します。
4. [削除] を選択します。
5. [マルチリージョンアクセスポイントの削除] ダイアログボックスで、削除する AWS バケットの名前を入力します。

   > **Note**
   >
   > 有効なバケット名を入力していることを確認してください。有効ではない場合、[削除] ボタンが無効になります。

6. [削除] を選択して、マルチリージョンアクセスポイントの削除を確認します。

AWS CLI の使用

AWS CLI を使用して、マルチリージョンアクセスポイントを削除できます。このアクションでは、マルチリージョンアクセスポイントに関連付けられたバケットは削除されず、マルチリージョンアクセスポイント自体だけが削除されます。このコマンドの例を実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control delete-multi-region-access-point --account-id 123456789012 --details
 Name=example-multi-region-access-point-name
```

AWS PrivateLink で使用するマルチリージョンアクセスポイントの設定

マルチリージョンアクセスポイントを使用して、AWS リージョン 間で Amazon S3 リクエストトラフィックのルーティングを行うことができます。各マルチリージョンアクセスポイントのグローバルエンドポイントは、複数の送信元からの Amazon S3 データリクエストトラフィックをルーティン

グします。個別のエンドポイントで複雑なネットワーク設定を構築する必要はありません。これらのデータリクエストトラフィックソースには以下が含まれます。

- 仮想プライベートクラウド (VPC) から発信されるトラフィック
- AWS PrivateLink を経由する、オンプレミスのデータセンターからのトラフィック
- パブリックインターネットからのトラフィック

S3 マルチリージョンアクセスポイントへの AWS PrivateLink 接続を確立すると、シンプルなネットワークアーキテクチャと設定を使用して、AWS に、またはプライベート接続を経由して複数の AWS リージョン に、S3 リクエストをルーティングできます。AWS PrivateLink を使用する場合、VPC ピアリング接続を設定する必要はありません。

トピック

- AWS PrivateLink で使用するマルチリージョンアクセスポイントの設定
- VPC エンドポイントからマルチリージョンアクセスポイントへのアクセスを削除する

AWS PrivateLink で使用するマルチリージョンアクセスポイントの設定

AWS PrivateLink は、Virtual Private Cloud (VPC) 内のプライベート IP アドレスを使用して Amazon S3 にプライベート接続できます。VPC 内で 1 つ以上のインターフェイスエンドポイントをプロビジョニングして、Amazon S3 マルチリージョンアクセスポイントに接続できます。

AWS Management Console、AWS CLI、または AWS SDK によって、マルチリージョンアクセスポイントの com.amazonaws.s3-global.accesspoint エンドポイントを作成することができます。マルチリージョンアクセスポイントのインターフェイスエンドポイントを設定する方法の詳細については、VPC ユーザーガイドの「インターフェイス VPC エンドポイント」を参照してください。

インターフェイスエンドポイント経由でマルチリージョンアクセスポイントにリクエストを送信するには、次の手順に従って VPC とマルチリージョンアクセスポイントを設定します。

AWS PrivateLink で使用するマルチリージョンアクセスポイントを設定するには

1. マルチリージョンアクセスポイントに接続できる適切な VPC エンドポイントを作成するか、所有します。VPC エンドポイントの作成の詳細については、VPC ユーザーガイドの「インターフェイス VPC エンドポイント 」を参照してください。

> **Important**
>
> 必ず com.amazonaws.s3-global.acesspoint エンドポイントを作成してください。他のエンドポイントタイプでは、マルチリージョンアクセスポイントにアクセスできません。

この VPC エンドポイントが作成されると、エンドポイントに対してプライベート DNS が有効になっている場合、VPC 内のすべてのマルチリージョンアクセスポイントリクエストはこのエンドポイントを経由します。これはデフォルトで有効になっています。

2. マルチリージョンアクセスポイントポリシーが VPC エンドポイントからの接続をサポートしていない場合は、更新する必要があります。
3. 個々のバケットポリシーで、マルチリージョンアクセスポイントのユーザーへのアクセスが許可されていることを確認します。

マルチリージョンアクセスポイントは、リクエスト自体を満たすのではなく、バケットにリクエストをルーティングすることによって機能することに注意してください。リクエストの発信者は、マルチリージョンアクセスポイントに対する許可を持ち、マルチリージョンアクセスポイント内の個々のバケットへのアクセスが許可されている必要があることを覚えておくことが重要です。そうしないと、リクエストは、発信者がリクエストを実行するための許可を持たないバケットにルーティングされる可能性があります。マルチリージョンアクセスポイントと関連付けられているバケットは、同じまたは別の AWS アカウントに所有されることができます。ただし、許可が正しく設定されていれば、異なるアカウントの VPC でマルチリージョンアクセスポイントを使用できます。

このため、VPC エンドポイントポリシーでは、マルチリージョンアクセスポイントと、リクエストを処理できるようにする基盤となる各バケットの両方へのアクセスを許可する必要があります。例えば、エイリアス `mfzwi23gnjvgw.mrap` を持つマルチリージョンアクセスポイントがあるとします。バケット `DOC-EXAMPLE-BUCKET1` および `DOC-EXAMPLE-BUCKET2` によって支えられており、すべて AWS アカウント `123456789012` が所有しています。この場合、以下の VPC エンドポイントでは、`mfzwi23gnjvgw.mrap` に対して行われた VPC からの `GetObject` リクエストを、いずれかのバッキングバケットで実行できます。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Sid": "Read-buckets-and-MRAP-VPCE-policy",
        "Principal": "*",
```

```
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*",
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET2/*",
            "arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap/object/*"
        ]
    }]
}
```

前述したように、マルチリージョンアクセスポイントポリシーが VPC エンドポイント経由のアクセスをサポートするように設定されていることも確認する必要があります。アクセスをリクエストしている VPC エンドポイントを指定する必要はありません。次のサンプルポリシーは、GetObject リクエストにマルチリージョンアクセスポイントを使用しようとしているリクエスターにアクセスを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Sid": "Open-read-MRAP-policy",
        "Effect": "Allow",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
          ],
        "Resource": "arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap/object/*"
    }]
}
```

もちろん、個々のバケットには、VPC エンドポイントを介して送信されたリクエストからのアクセスをサポートするポリシーが必要です。以下のポリシーの例では、VPC エンドポイントを介して行われたリクエストを含む匿名ユーザーに読み取りアクセス許可が付与されます。

```
{
    "Version":"2012-10-17",
   "Statement": [
   {
       "Sid": "Public-read",
       "Effect":"Allow",
```

```
        "Principal": "*",
        "Action": "s3:GetObject",
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET1",
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET2/*"]
    }]
}
```

VPC エンドポイントポリシーの編集については、「VPC ユーザーガイド」の「VPC エンドポイントによるサービスへのアクセスのコントロール」を参照してください。

VPC エンドポイントからマルチリージョンアクセスポイントへのアクセスを削除する

マルチリージョンアクセスポイントを所有しており、インターフェイスエンドポイントからそのアクセス権を削除する場合は、マルチリージョンアクセスポイントに新しいアクセスポリシーを提供して、VPC エンドポイントを経由するリクエストへのアクセスを防止する必要があります。ただし、マルチリージョンアクセスポイントのバケットが VPC エンドポイント経由のリクエストをサポートしている場合、これらのリクエストは引き続きサポートされます。このサポートを禁止する場合は、バケットのポリシーも更新しなければなりません。マルチリージョンアクセスポイントに新しいアクセスポリシーを指定すると、マルチリージョンアクセスポイントへのアクセスだけが禁止され、基になるバケットへのアクセスは禁止されません。

> **Note**
>
> マルチリージョンアクセスポイントのアクセスポリシーを削除することはできません。マルチリージョンアクセスポイントへのアクセスを削除するには、変更したアクセスで新しいアクセスポリシーを提供する必要があります。

マルチリージョンアクセスポイントのアクセスポリシーを更新する代わりに、バケットポリシーを更新して、VPC エンドポイントを使用したリクエストを防ぐことができます。この場合、ユーザーは VPC エンドポイント経由でマルチリージョンアクセスポイントにアクセスできます。ただし、マルチリージョンアクセスポイントリクエストが、バケットポリシーがアクセスを禁止するバケットにルーティングされると、リクエストによりエラーメッセージが生成されます。

マルチリージョンアクセスポイントを使用したリクエスト

他のリソースと同様に、Amazon S3 マルチリージョンアクセスポイントには、Amazon リソースネーム (ARN) があります。これらの ARN を使用することで、AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 API を使用してマルチリージョンのアクセスポイントにリクエストを指示できます。これらの ARN を使用して、アクセスコントロールポリシーでマルチリージョンアクセスポイントを識別することもできます。マルチリージョンアクセスポイント ARN にマルチリージョンアクセスポイントの名前が含まれたり、公開されたりすることはありません。ARN の詳細については、「AWS 全般のリファレンス」の「Amazon リソースネーム (ARN)」を参照してください。

> **Note**
>
> マルチリージョンアクセスポイントのエイリアスと ARN は互換的に使用できません。

マルチリージョンアクセスポイント ARN は、以下の形式を使用します。

`arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`

以下に、マルチリージョンアクセスポイント ARN の例をいくつか示します。

- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap` は、AWS アカウント 123456789012 が所有する、エイリアス `mfzwi23gnjvgw.mrap` を持つマルチリージョンアクセスポイントを表します。
- `arn:aws:s3::123456789012:accesspoint/*` は、アカウント 123456789012 内のすべてのマルチリージョンアクセスポイントを表します。この ARN は、アカウント 123456789012 のすべてのマルチリージョンアクセスポイントと一致しますが、ARN に AWS リージョン が含まれていないため、リージョンの Amazon S3 Access Points とは一致しません。これとは対照的に、ARN `arn:aws:s3:us-west-2:123456789012:accesspoint/*` は、リージョン `us-west-2` にあるアカウント 123456789012 のすべてのリージョンの Amazon S3 Access Points とは一致しますが、マルチリージョンアクセスポイントとは一致しません。

マルチリージョンアクセスポイントを介してアクセスされるオブジェクトの ARN は、以下の形式を使用します。

`arn:aws:s3::account_id:accesspoint/MultiRegionAccessPoint_alias//key`

マルチリージョンアクセスポイント ARN と同様に、マルチリージョンアクセスポイントを介してアクセスされるオブジェクトの ARN には AWS リージョン は含まれません。次に例を示します。

- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//-01` は、アカウント `123456789012` が所有する、エイリアス `mfzwi23gnjvgw.mrap` を持つマルチリージョンアクセスポイントを通じてアクセスされる、`-01` を表します。
- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//*` は、アカウント `123456789012` で、エイリアス `mfzwi23gnjvgw.mrap` を持ち、マルチリージョンアクセスポイントを通じてアクセスできるすべてのオブジェクトを表します。
- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//-01/finance/*` は、アカウント `123456789012` で、エイリアス `mfzwi23gnjvgw.mrap` を持ち、マルチリージョンアクセスポイントの `-01/finance/` でアクセスできるすべてのオブジェクトを表します。

マルチリージョンアクセスポイントのホスト名

マルチリージョンアクセスポイントのホスト名を使用して、マルチリージョンアクセスポイントを介して Amazon S3 のデータにアクセスできます。パブリックインターネットからこのホスト名にリクエストを送信できます。マルチリージョンアクセスポイントに 1 つ以上のインターネットゲートウェイを設定している場合は、仮想プライベートクラウド (VPC) からこのホスト名にリクエストを送信することもできます。マルチリージョンアクセスポイントで使用する VPC インターフェイスエンドポイントの作成の詳細については、「AWS PrivateLink で使用するマルチリージョンアクセスポイントの設定」を参照してください。

また、VPC エンドポイントを使用して VPC からマルチリージョンアクセスポイント経由でリクエストを行うには、AWS PrivateLink を使用することができます。AWS PrivateLink を使用してマルチリージョンアクセスポイントへのリクエストを行う場合、*`region`*`.vpce.amazonaws.com` で終わるエンドポイント固有のリージョンのドメインネームシステム (DNS) を直接使用することはできません。このホスト名には証明書が関連付けられていないため、直接使用することはできません。ただし、引き続き、VPC エンドポイントのパブリックドメインネームシステム (DNS) 名を `CNAME` または `ALIAS` ターゲットとして使用することはできます。あるいは、エンドポイントでプライベートドメインネームシステム (DNS) を有効にし、このセクションで説明されているように、標準のマルチリージョンアクセスポイント *`MultiRegionAccessPoint_alias`*`.accesspoint.s3-global.amazonaws.com` ドメインネームシステム (DNS) 名を使用することもできます。

マルチリージョンアクセスポイント経由で API for Amazon S3 データオペレーション (例えば `GetObject`) にリクエストする場合、リクエストのホスト名は次のようになります。

MultiRegionAccessPoint_alias.accesspoint.s3-global.amazonaws.com

例えば、エイリアス mfzwi23gnjvgw.mrap を持つマルチリージョンアクセスポイントを介して GetObject リクエストを行うには、ホスト名 mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com に対してクエストを行います。ホスト名の s3-global 部分は、このホスト名が特定のリージョン用ではないことを示しています。

マルチリージョンアクセスポイントを使用したリクエストの作成は、単一リージョンアクセスポイントを使用したリクエストの作成と似ています。ただし、以下の違いを考慮する必要があります。

- マルチリージョンアクセスポイント ARN に AWS リージョン は含まれません。フォーマット arn:aws:s3::*account-id*:accesspoint/*MultiRegionAccessPoint_alias* に従います。
- API オペレーションを介して行われたリクエスト (これらのリクエストに ARN を使用する必要はありません) では、マルチリージョンアクセスポイントは異なるエンドポイントスキームを使用します。スキームは *MultiRegionAccessPoint_alias*.accesspoint.s3-global.amazonaws.com です。例えば、mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com です。単一リージョンアクセスポイントとの相違点に注意してください。
  - マルチリージョンアクセスポイントのホスト名は、マルチリージョンアクセスポイント名ではなく、エイリアスを使用します。
  - マルチリージョンアクセスポイントのホスト名には、所有者の AWS アカウント ID は含まれません。
  - マルチリージョンアクセスポイントのホスト名には、AWS リージョン は含まれません。
  - マルチリージョンアクセスポイントのホスト名には、s3.amazonaws.com の代わりに s3-global.amazonaws.com が含まれます。
- マルチリージョンアクセスポイントのリクエストは、署名バージョン 4A (SigV4A) を使用して署名する必要があります。AWS SDK を使用時する場合、SDK は Sigv4 を SigV4A に自動的に変換します。そのため、ご使用の AWS SDK が、グローバル AWS リージョン リクエストの署名に使用される署名実装として SigV4AA をサポートしていることを確認してください。SigV4A の詳細については、「AWS 全般のリファレンス」の「AWS API リクエストの署名」を参照してください。

マルチリージョンアクセスポイントと Amazon S3 Transfer Acceleration

Amazon S3 Transfer Acceleration は、バケットへの高速データ転送を実現する機能です。Transfer Acceleration は個々のバケットレベルで設定されます。Transfer Acceleration の詳細については、「Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定」を参照してください。

マルチリージョンアクセスポイントは、Transfer Acceleration と同様の高速転送メカニズムを使用して、大きなオブジェクトを AWS ネットワークで送信します。このため、マルチリージョンアクセスポイント経由でリクエストを送信する場合、Transfer Acceleration を使用する必要はありません。この転送パフォーマンスの向上は、マルチリージョンアクセスポイントに自動的に組み込まれます。

トピック

許可

Amazon S3 マルチリージョンアクセスポイントは、複数の AWS リージョン の Amazon S3 バケットへのデータアクセスを簡素化します。マルチリージョンアクセスポイントは、Amazon S3 のデータアクセスオブジェクトのオペレーション (`GetObject` や `PutObject` など) を実行するために使用できる名前付きのグローバルエンドポイントです。各マルチリージョンアクセスポイントは、グローバルエンドポイントを介したすべてのリクエストについて、個別のアクセス許可とネットワークコントロールを設定できます。

各マルチリージョンアクセスポイントは、基になるバケットにアタッチされたバケットポリシーと連動して機能するカスタマイズされたアクセスポリシーを適用することもできます。リクエストが成功するには、以下のすべてでオペレーションが許可されている必要があります。

- マルチリージョンアクセスポイントポリシー

- 基になる AWS Identity and Access Management (IAM) ポリシー
- 基になるバケットポリシー (リクエストのルーティング先)

マルチリージョンアクセスポイントポリシーは、特定の IAM ユーザーまたはグループからのリクエストのみを受け入れるように設定できます。これを行う方法の例については、「the section called "マルチリージョンアクセスポイントポリシーの例"」の例 2 を参照してください。Virtual Private Cloud (VPC) からのリクエストだけを受け入れるようにマルチリージョンアクセスポイントポリシーを設定することで、プライベートネットワークへの Amazon S3 データアクセスを制限できます。

例えば、次のようになります。`AppDataReader` というユーザーを AWS アカウントで使用して、マルチリージョンアクセスポイントを介して `GetObject` リクエストを行うとします。リクエストが拒否されないようにするために、マルチリージョンアクセスポイントと、マルチリージョンアクセスポイントの基盤となる各バケットによって `AppDataReader` ユーザーに `s3:GetObject` 許可を付与する必要があります。`AppDataReader` は、この許可を付与しないバケットからデータを取得できません。

> **Important**
>
> バケットのアクセスコントロールをマルチリージョンアクセスポイントポリシーに委任しても、バケット名または Amazon リソースネーム (ARN) を使用してバケットに直接アクセスする場合のバケットの動作は変わりません。バケットに対して直接行われるすべてのオペレーションは、以前と同じように動作します。マルチリージョンアクセスポイントポリシーに含めた制限は、そのマルチリージョンアクセスポイントを介したリクエストにのみ適用されます。

マルチリージョンアクセスポイントへのパブリックアクセスの管理

マルチリージョンアクセスポイントは、 マルチリージョンアクセスポイントごとに独立したブロックパブリックアクセス設定をサポートしています。 マルチリージョンアクセスポイントを作成するときに、その マルチリージョンアクセスポイントに適用するブロックパブリックアクセス設定を指定できます。

> **Note**
>
> マルチリージョンアクセスポイントの個別のパブリックアクセスをブロックする設定が無効になっていても、(自分のアカウントの) [このアカウントのブロックパブリックアクセス設定] または [外部バケットのブロックパブリック設定] は引き続き適用されます。

マルチリージョンアクセスポイントを介して行われたリクエストについて、Amazon S3 は以下に対するブロックパブリックアクセス設定を評価します。

- マルチリージョンアクセスポイント
- 基になるバケット (外部バケットを含む)
- マルチリージョンアクセスポイントを所有するアカウント
- 基になるバケットを所有するアカウント (外部アカウントを含む)

これらの設定のいずれかで、リクエストをブロックする必要があることが示されると、Amazon S3 はリクエストを拒否します。Amazon S3 ブロックパブリックアクセス機能の詳細については、「Amazon S3 ストレージへのパブリックアクセスのブロック」を参照してください。

> **Important**
>
> デフォルトでは、マルチリージョンアクセスポイントに対してすべてのブロックパブリックアクセス設定が有効になります。 マルチリージョンアクセスポイントに適用したくない設定がある場合は、明示的に無効にする必要があります。
> マルチリージョンアクセスポイントを作成した後にブロックパブリックアクセス設定を変更することはできません。

マルチリージョンアクセスポイントのパブリックアクセスブロック設定の表示

マルチリージョンアクセスポイントのパブリックアクセスブロック設定を表示するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Multi-Region Access Points] (マルチリージョンアクセスポイント) を選択します。

3. 確認するマルチリージョンアクセスポイントの名前を選択します。
4. [アクセス許可] タブを選択します。
5. [Block Public Access settings for this Multi-Region Access Point] (このマルチリージョンアクセスポイントのパブリックアクセス設定をブロック) で、マルチリージョンアクセスポイントのブロックパブリックアクセス設定を確認します。

> **Note**
>
> マルチリージョンアクセスポイントを作成した後、ブロックパブリックアクセス設定を編集することはできません。したがって、パブリックアクセスをブロックする場合は、マルチリージョンアクセスポイントを作成する前に、パブリックアクセスがなくてもアプリケーションが正しく動作することを確認してください。

マルチリージョンアクセスポイントポリシーの使用

次のマルチリージョンアクセスポイントポリシーの例では、IAM ユーザーに対して、マルチリージョンアクセスポイントのファイルを一覧表示およびダウンロードするアクセス権限を付与しています。このポリシーの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
  "Version":"2012-10-17",
  "Statement":[
     {
        "Effect":"Allow",
        "Principal":{
           "AWS":"arn:aws:iam::123456789012:user/JohnDoe"
        },
        "Action":[
           "s3:ListBucket",
           "s3:GetObject"
        ],
        "Resource":[
           "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
           "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/object/
*"
        ]
     }
  ]
```

```
}
```

AWS Command Line Interface (AWS CLI) を使用して、マルチリージョンアクセスポイントポリシーを指定されたマルチリージョンアクセスポイントに関連付けるには、次の put-multi-region-access-point-policy コマンドを使用します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。各マルチリージョンアクセスポイントに設定できるポリシーは 1 つだけなので、put-multi-region-access-point-policy アクションをリクエストすると、指定されたマルチリージョンアクセスポイントに関連付けられている既存のポリシーがすべて置き換えられます。

AWS CLI

```
aws s3control put-multi-region-access-point-policy
--account-id 111122223333
--details { "Name": "DOC-EXAMPLE-BUCKET-MultiRegionAccessPoint",
 "Policy": "{ \"Version\": \"2012-10-17\", \"Statement\": { \"Effect\":
 \"Allow\", \"Principal\": { \"AWS\": \"arn:aws:iam::111122223333:root
\" }, \"Action\": [\"s3:ListBucket\", \"s3:GetObject\"], \"Resource\":
 [ \"arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
 \"arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/object/*
\" ] } }" }
```

前のオペレーションの結果をクエリするには、次のコマンドを使用します。

AWS CLI

```
aws s3control describe-multi-region-access-point-operation
--account-id 111122223333
--request-token-arn requestArn
```

マルチリージョンアクセスポイントポリシーを取得するには、次のコマンドを使用します。

AWS CLI

```
aws s3control get-multi-region-access-point-policy
--account-id 111122223333
--name=DOC-EXAMPLE-BUCKET-MultiRegionAccessPoint
```

マルチリージョンアクセスポイントポリシーの編集

マルチリージョンアクセスポイントポリシー (JSON で記述) は、このマルチリージョンアクセスポイントで使用する Amazon S3 バケットへのストレージアクセスを提供します。特定のプリンシパルがマルチリージョンアクセスポイントでさまざまなアクションを実行することを許可または拒否できます。リクエストがマルチリージョンアクセスポイントを介してバケットにルーティングされると、マルチリージョンアクセスポイントとバケットの両方のアクセスポリシーが適用されます。より制限の厳しいアクセスポリシーが常に優先されます。

> **Note**
>
> バケットに他のアカウントが所有するオブジェクトが含まれている場合、マルチリージョンアクセスポイントのポリシーは他の AWS アカウント が所有するオブジェクトには適用されません。

マルチリージョンアクセスポイントポリシーを適用した後、ポリシーを削除することはできません。ポリシーを編集するか、既存のポリシーを上書きする新しいポリシーを作成することができます。

マルチリージョンアクセスポイントポリシーを編集するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Multi-Region Access Points] (マルチリージョンアクセスポイント) を選択します。
3. ポリシーを編集するマルチリージョンアクセスポイントの名前を選択します。
4. [アクセス許可] タブを選択します。
5. [Multi-Region Access Point policy] (マルチリージョンアクセスポイントポリシー) セクションまでスクロールします。[Edit] (編集) を選択し、ポリシーを更新します。
6. [Edit Multi-Region Access Point policy] (マルチリージョンアクセスポイントポリシーの編集) ページが表示されます。ポリシーをテキストフィールドに直接入力するか、[Add statement] (ステートメントの追加) を選択してドロップダウンリストからポリシーの要素を選択できます。

   > **Note**
   >
   > コンソールに、マルチリージョンアクセスポイントの Amazon リソースネーム (ARN) が自動的に表示されます。これをポリシーで使用できます。マルチリージョンアクセス

ポイントポリシーの例については、「the section called “マルチリージョンアクセスポイントポリシーの例”」を参照してください。

マルチリージョンアクセスポイントポリシーの例

Amazon S3 マルチリージョンアクセスポイントは、AWS Identity and Access Management (IAM) リソースポリシーをサポートします。これらのポリシーを使用して、リソース、ユーザー、その他の条件別にマルチリージョンアクセスポイントの使用を制御することができます。アプリケーションやユーザーがマルチリージョンアクセスポイントを介してオブジェクトにアクセスできるようにするには、マルチリージョンアクセスポイントと基になるバケットの両方が同じアクセスを許可している必要があります。

マルチリージョンアクセスポイントと基になるバケットの両方に同じアクセスを許可するには、次のいずれかを実行します。

- (推奨) Amazon S3 マルチリージョンアクセスポイントを使用する際のアクセスコントロールを簡素化するには、Amazon S3 バケットのアクセスコントロールをマルチリージョンアクセスポイントに委任します。これを行う方法の例については、このセクションの例 1 を参照してください。
- マルチリージョンアクセスポイントポリシーに含まれているものと同じアクセス許可を、基になるバケットのポリシーに追加します。

Important

バケットのアクセスコントロールをマルチリージョンアクセスポイントポリシーに委任しても、バケット名または Amazon リソースネーム (ARN) を使用してバケットに直接アクセスする場合のバケットの動作は変わりません。バケットに対して直接行われるすべてのオペレーションは、以前と同じように動作します。マルチリージョンアクセスポイントポリシーに含めた制限は、そのマルチリージョンアクセスポイントを介したリクエストにのみ適用されます。

Example 1 - (同じアカウントまたはクロスアカウントの) バケットポリシーで特定のマルチリージョンアクセスポイントへのアクセスを委任する

以下のバケットポリシーの例では、特定のマルチリージョンアクセスポイントにフルバケットアクセスを付与しています。したがって、このバケットへのすべてのアクセスは、そのマルチリージョンアクセスポイントに添付されているポリシーによってコントロールされます。バケットへの直接アクセ

スを必要としないすべてのユースケースでは、この方法でバケットを設定することをお勧めします。このバケットポリシー構造は、同じアカウントまたは別のアカウントのマルチリージョンアクセスポイントに使用できます。

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointArn" : "MultiRegionAccessPoint_ARN" }
        }
    }]
}
```

> **Note**
>
> アクセス権を付与する複数のマルチリージョンアクセスポイントがある場合は、必ず個々のマルチリージョンアクセスポイントを一覧表示してください。

Example 2 – マルチリージョンアクセスポイントポリシーでマルチリージョンアクセスポイントにアカウントアクセスを付与する

以下のマルチリージョンアクセスポイントポリシーでは、*MultiRegionAccessPoint_ARN* によって定義されたマルチリージョンアクセスポイントに含まれるオブジェクトを表示および読み取る *123456789012* アクセス許可をアカウントに付与します。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
       "Effect": "Allow",
       "Principal": {
          "AWS":"arn:aws:iam::123456789012:user/JohnDoe"
       },
       "Action":[
          "s3:ListBucket",
```

```
            "s3:GetObject"
         ],
         "Resource":[
            "MultiRegionAccessPoint_ARN",
            "MultiRegionAccessPoint_ARN/object/*"
         ]
      }
   ]
}
```

Example 3 – バケットのリスト取得を許可するマルチリージョンアクセスポイントポリシー

以下のマルチリージョンアクセスポイントポリシーでは、*MultiRegionAccessPoint_ARN* によって定義されたマルチリージョンアクセスポイントに含まれるオブジェクトを表示する *123456789012* アクセス許可をアカウントに付与します。

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
       "Effect": "Allow",
       "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/JohnDoe"
        },
        "Action": "s3:ListBucket",
        "Resource": "MultiRegionAccessPoint_ARN"
       }
    ]
}
```

マルチリージョンアクセスポイントの制約および制限事項

Amazon S3 のマルチリージョンアクセスポイントには以下の制約と制限事項があります。

- マルチリージョンアクセスポイント名:
  - 1 つの AWS アカウント内で一意である
  - 数字または小文字で始める
  - 3 ~ 50 文字の長さにする
  - 名前をハイフン (-) で開始または終了することはできません。
  - 下線 (_)、大文字、ピリオド (.) は使用しない

  - 作成後は編集できない
- マルチリージョンアクセスポイントのエイリアスは Amazon S3 によって生成され、編集や再利用はできません。
- ゲートウェイエンドポイントを使用して、マルチリージョンアクセスポイントを介してデータにアクセスすることはできません。ただし、インターフェイスエンドポイントを使用して、マルチリージョンアクセスポイントを介してデータにアクセスすることはできます。AWS PrivateLink を使用するには、VPC エンドポイントを作成する必要があります。詳細については、「AWS PrivateLink で使用するマルチリージョンアクセスポイントの設定」を参照してください。
- Amazon CloudFront でマルチリージョンアクセスポイントを使用するには、マルチリージョンアクセスポイントを `Custom Origin` ディストリビューションタイプとして設定する必要があります。さまざまなオリジンタイプの詳細については、「CloudFront ディストリビューションでのさまざまなオリジンの使用」を参照してください。Amazon CloudFront でのマルチリージョンアクセスポイントの使用については、AWS ストレージブログの「Building an active-active, proximity-based application across multiple Regions」を参照してください。
- マルチリージョンアクセスポイントの最小要件：
  - Transport Layer Security (TLS) v1.2
  - 署名バージョン 4 (SigV4A)

    マルチリージョンアクセスポイントは、署名バージョン 4A をサポートします。このバージョンの SigV4 では、複数の AWS リージョン についてリクエストに署名できます。この機能は、いくつかのリージョンのいずれかからデータにアクセスする可能性がある API 操作に役立ちます。AWS SDK を使用する場合、認証情報を提供すると、マルチリージョンアクセスポイントへのリクエストでは、追加の設定なしで署名バージョン 4A が使用されます。必ず SigV4A アルゴリズムと AWS SDK との互換性を確認してください。SigV4A の詳細については、「AWS 全般のリファレンス」の「AWS API リクエストの署名」を参照してください。

    > **Note**
    >
    > 例えば AWS Identity and Access Management IAM ロールを使用する場合など、一時的なセキュリティ認証情報で SigV4A を使用するには、必ずリージョン AWS Security Token Service (AWS STS) でエンドポイントから一時的認証情報を要求してください。グローバル AWS STS エンドポイント (`sts.amazonaws.com`) から一時的な認証情報をリクエストする場合は、まず、グローバルエンドポイントに対するセッショントークンのリージョンの互換性をすべての AWS リージョンで有効に設定する必要があります。

詳細については、IAM ユーザーガイドの 「AWS リージョン での AWS STS の管理」を参照してください。

- マルチリージョンアクセスポイントは、匿名リクエストをサポートしていません。
- マルチリージョンアクセスポイントの制限事項：
  - IPv6 はサポートされていません。
  - Amazon S3 on Outposts バケットはサポートされていません。
  - マルチリージョンアクセスポイントは、マルチリージョンアクセスポイント ARN を使用する場合に限り、マルチリージョンアクセスポイントを宛先とするコピー操作をサポートします。
  - S3 バッチオペレーション機能はサポートされていません。
- 一部の AWS SDK はサポートされていません。マルチリージョンアクセスポイントでどの AWS SDK がサポートされているかを確認するには、「AWS SDK との互換性」を参照してください。
- マルチリージョンアクセスポイントの Service Quotas は、以下のとおりです。
  - アカウントごとに、最大 100 のマルチリージョンアクセスポイントです。
  - 1 つのマルチリージョンアクセスポイントには、17 リージョンの制約があります。
- マルチリージョンアクセスポイントの作成後は、マルチリージョンアクセスポイント設定でのバケットの追加、変更、または削除はできません。バケットを変更するには、マルチリージョンアクセスポイント全体を削除し、新しいリージョンアクセスポイントを作成する必要があります。マルチリージョンアクセスポイントのクロスアカウントバケットが削除された場合、このバケットを再接続する唯一の方法は、そのアカウントで同じ名前とリージョンを使用してバケットを再作成することです。
- マルチリージョンアクセスポイントで使用されている (同じアカウントにある) 基になるバケットは、マルチリージョンアクセスポイントを削除した後でのみ削除できます。
- マルチリージョンアクセスポイントの作成または保守のすべてのコントロールプレーンリクエストは、US West (Oregon) リージョンにルーティングされる必要があります。マルチリージョンアクセスポイントのデータプレーンリクエストでは、リージョンを指定する必要はありません。
- マルチリージョンアクセスポイントのフェイルオーバーコントロールプレーンの場合、リクエストは次の 5 つのサポートされているリージョンのいずれかにルーティングされる必要があります。
  - US East (N. Virginia)
  - US West (Oregon)
  - Asia Pacific (Sydney)
  - Asia Pacific (Tokyo)
  - Europe (Ireland)

- マルチリージョンアクセスポイントは、以下の AWS リージョン のバケットのみをサポートします。
  - US East (N. Virginia)
  - US East (Ohio)
  - US West (N. California)
  - US West (Oregon)
  - Asia Pacific (Mumbai)
  - Asia Pacific (Osaka)
  - Asia Pacific (Seoul)
  - Asia Pacific (Singapore)
  - Asia Pacific (Sydney)
  - Asia Pacific (Tokyo)
  - Canada (Central)
  - Europe (Frankfurt)
  - Europe (Ireland)
  - Europe (London)
  - Europe (Paris)
  - Europe (Stockholm)
  - South America (São Paulo)

マルチリージョンアクセスポイントリクエストのルーティング

マルチリージョンアクセスポイントを介してリクエストを行うと、Amazon S3 は、マルチリージョンアクセスポイントに関連付けられたバケットのうち、どのバケットが最も近くにあるかを決定します。Amazon S3 は、リクエストをそのバケットに送信します。これは、配置されている AWS リージョンに関わりません。

マルチリージョンアクセスポイントが最も近くにあるバケットにリクエストをルーティングすると、Amazon S3 はリクエストをそのバケットに直接送信したかのように処理します。マルチリージョンアクセスポイントは、Amazon S3 バケットのデータコンテンツを認識しません。そのため、リクエストを受け取ったバケットには、リクエストされたデータが含まれていない可能性があります。マルチリージョンアクセスポイントに関連付けられた Amazon S3 バケットに一貫性のあるデー

タセットを作成するには、S3 クロスリージョンレプリケーション (CRR) を設定できます。その後、どのバケットも リクエストを正常に処理します。

Amazon S3 は、次のルールに従って、マルチリージョンアクセスポイントリクエストを送信します。

- Amazon S3 は、近接の度合いに応じてリクエストが実行されるように最適化します。マルチリージョンアクセスポイントでサポートされているバケットを調べて、最も近いバケットにリクエストをリレーします。
- リクエストが既存のリソース (例えば、`GetObject`) を指定しても、Amazon S3 は、リクエストを実行するときにオブジェクトの名前を考慮しません。つまり、マルチリージョンアクセスポイントの 1 つのバケットにオブジェクトが存在する場合でも、リクエストはそのオブジェクトを含まないバケットにルーティングできるということです。この状況では、クライアント に 404 エラーメッセージが返されます。

  404 エラーを避けるには、バケットに S3 クロスリージョンレプリケーション (CRR) を設定することをお勧めします。レプリケーションにより、目的のオブジェクトがマルチリージョンアクセスポイントのバケットには存在するが、リクエストがルーティングされた特定のバケットに存在しない場合に発生する可能性のある問題を解決できます。レプリケーション設定については、「マルチリージョンアクセスポイントで使用するためのレプリケーションの設定」を参照してください。

  必要な特定のオブジェクトを使用してリクエストが確実に受理されるようにするには、バケットのバージョニングを有効にし、リクエストにバージョン ID を含めることもお勧めします。このアプローチにより、探しているオブジェクトの正しいバージョンを確実に取得できます。バージョニングを有効にしたバケットでは、誤って上書きしても、オブジェクトを復旧できます。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。
- リクエストがリソースを作成するものである場合 (例えば、`PutObject` または `CreateMultipartUpload`)、Amazon S3 は最も近いバケットを使用してリクエストを実行します。例えば、世界中のどこからでも動画のアップロードをサポートしたい動画会社があるとします。ユーザーが PUT リクエストをマルチリージョンアクセスポイントに送信すると、オブジェクトは最も近くにあるバケットに配置されます。アップロードした動画を世界中の他のユーザーが最短のレイテンシーでダウンロードできるようにするには、CRR と双方向レプリケーションを使用できます。CRR を双方向レプリケーションと使用すると、マルチリージョンアクセスポイントに関連付けられたすべてのバケットのコンテンツを同期された状態に維持します。マルチリージョンアクセスポイントでのレプリケーションの使用の詳細については、「マルチリージョンアクセスポイントで使用するためのレプリケーションの設定」を参照してください。

Amazon S3 マルチリージョンアクセスポイントフェイルオーバーコントロール

Amazon S3 マルチリージョンアクセスポイントのフェイルオーバーコントロールを使用すると、リージョン別トラフィックが中断してもビジネス継続性を維持できると同時に、アプリケーションをマルチリージョンアーキテクチャにしてコンプライアンスと冗長性のニーズを満たすことができます。リージョン別トラフィックが中断された場合は、マルチリージョンアクセスポイントのフェイルオーバーコントロールを使用して、Amazon S3 マルチリージョンアクセスポイントの背後にあるどの AWS リージョン がデータアクセスとストレージのリクエストを処理するかを選択できます。

フェイルオーバーをサポートするには、マルチリージョンアクセスポイントをアクティブ/パッシブ構成に設定できます。トラフィックは通常の状態ではアクティブリージョンに流れ、パッシブリージョンはフェイルオーバー用にスタンバイになります。

例えば、選択した AWS リージョン へのフェイルオーバーを行うには、プライマリ (アクティブ) リージョンからのトラフィックをセカンダリ (パッシブ) リージョンに移行します。このようなアクティブ/パッシブ構成では、一方のバケットはアクティブでトラフィックを受け入れますが、もう一方のバケットはパッシブでトラフィックを受け入れません。パッシブバケットは災害対策に使用されます。フェイルオーバーを開始すると、すべてのトラフィック (GET や PUT リクエストなど) はアクティブ状態 (1 つのリージョン内) のバケットに送られ、パッシブ状態のバケット (別のリージョン内) から離れます。

S3 クロスリージョンレプリケーション (CRR) を双方向のレプリケーションルールで有効にしている場合、フェイルオーバー中もバケットを同期された状態にしておくことができます。さらに、アクティブ/アクティブ設定で CRR を有効にしている場合、Amazon S3 マルチリージョンアクセスポイントは、最も近いバケットの場所からデータを取得できるため、アプリケーションのパフォーマンスが向上します。

AWS リージョン サポート

Amazon S3 マルチリージョンアクセスポイントのフェイルオーバーコントロールにより、S3 バケットはマルチリージョンアクセスポイントがサポートされている 17 リージョンのいずれにも存在することができます。フェイルオーバーは、任意の 2 つのリージョンで一度に開始できます。

> **Note**
>
> フェイルオーバーは一度に 2 つのリージョン間でのみ開始されますが、マルチリージョンアクセスポイントでは、同時に複数のリージョンのルーティングステータスを個別に更新できます。

以下のトピックでは、Amazon S3 マルチリージョンアクセスポイントフェイルオーバーコントロールの使用方法と管理方法について説明します。

トピック

Amazon S3 マルチリージョンアクセスポイントルーティング状態

Amazon S3 マルチリージョンアクセスポイントのフェイルオーバー設定は、マルチリージョンアクセスポイントで使用される AWS リージョン のルーティングステータスを決定します。Amazon S3 マルチリージョンアクセスポイントは、アクティブ/アクティブ状態またはアクティブ/パッシブ状態に設定できます。

- アクティブ/アクティブ — アクティブ/アクティブ設定では、すべてのリクエストはマルチリージョンアクセスポイントの最も近い AWS リージョン に自動的に送信されます。マルチリージョンアクセスポイントをアクティブ/アクティブ状態に設定すると、すべてのリージョンがトラフィックを受信できます。アクティブ/アクティブ設定でトラフィックの中断が発生した場合、ネットワークトラフィックはアクティブリージョンのいずれかに自動的にリダイレクトされます。
- アクティブ/パッシブ — アクティブ/パッシブ設定では、マルチリージョンアクセスポイントのアクティブリージョンはトラフィックを受信し、パッシブリージョンはトラフィックを受信しません。災害がある状況で S3 フェイルオーバーコントロールを使用してフェイルオーバーを開始する場合は、災害対策計画のテストと実行中に、マルチリージョンアクセスポイントをアクティブ/パッシブ設定にセットアップします。

Amazon S3 マルチリージョンアクセスポイントアクセスポイントフェイルオーバーコントロールの使用

このセクションでは、AWS Management Console を使用して、Amazon S3 マルチリージョンアクセスポイントフェイルオーバーコントロールを管理および使用する方法について説明します。

AWS Management Console のマルチリージョンアクセスポイントの詳細ページの [Failover configuration] (フェイルオーバーの設定) セクションには、[Edit routing status] (ルーティングステータスの編集) と[Failover] (フェイルオーバー) の 2 つのフェイルオーバーコントロールがあります。これらのコントロールは以下のように使用できます。

- [Edit routing status] (ルーティングステータスの編集) — [Edit routing status] (ルーティングステータスの編集) を選択すると、マルチリージョンアクセスポイントの 1 回のリクエストで最大 17 の AWS リージョン ルーティングステータスを手動で編集できます。[Edit routing status] (ルーティングステータスの編集) は次の目的で使用できます。
  - マルチリージョンアクセスポイントの 1 つ以上のリージョンのルーティングステータスを設定または編集する
  - 2 つのリージョンをアクティブ/パッシブ状態に設定してマルチリージョンアクセスポイントのフェイルオーバー設定を作成する
  - リージョンを手動でフェイルオーバーする
  - リージョン間でトラフィックを手動で切り替える
- [Failover] (フェイルオーバー) — [Failover] (フェイルオーバー) を選択してフェイルオーバーを開始すると、既にアクティブ/パッシブ状態に設定されている 2 つのリージョンのルーティングステータスが更新されるだけです。[Failover] (フェイルオーバー) を選択して開始したフェイルオーバー中に、2 つのリージョン間のルーティングステータスが自動的に切り替わります。

マルチリージョンアクセスポイントの、リージョンのルーティングステータスの編集

マルチリージョンアクセスポイントの詳細ページの [Failover configuration] (フェイルオーバー設定) セクションで [Edit routing status] (ルーティングステータスの編集) を選択すると、マルチリージョンアクセスポイントの 1 回のリクエストで最大 17 AWS リージョン のルーティングステータスを手動で更新できます。ただし、[Failover] (フェイルオーバー) を選択してフェイルオーバーを開始すると、既にアクティブ/パッシブ状態に設定されている 2 つのリージョンのルーティングステータスのみが更新されます。[Failover] (フェイルオーバー) を選択して開始したフェイルオーバーの間、2 つのリージョン間のルーティングステータスが自動的に切り替わります。

[Edit routing status] (ルーティングステータスの編集) (以下の手順で説明) は、次の目的で使用できます。

- マルチリージョンアクセスポイントの 1 つ以上のリージョンのルーティングステータスを設定または編集する
- 2 つのリージョンをアクティブ/パッシブ状態に設定してマルチリージョンアクセスポイントのフェイルオーバー設定を作成する
- リージョンを手動でフェイルオーバーする
- リージョン間でトラフィックを手動で切り替える

S3 コンソールの使用

マルチリージョンアクセスポイントのリージョンのルーティングステータスを更新するには

1. AWS マネジメントコンソールにサインインします。
2. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
3. ナビゲーションペインで、[Multi-Region Access Points] (マルチリージョンアクセスポイント) を選択します。
4. 更新するマルチリージョンアクセスポイントを選択します。
5. [Replication and failover] (レプリケーションとフェイルオーバー) タブを選択します。
6. ルーティングステータスを編集するリージョンを 1 つ以上選択します。

   > **Note**
   >
   > フェイルオーバーを開始するには、マルチリージョンアクセスポイントで少なくとも 1 つの AWS リージョン を [Active] (アクティブ) に、1 つのリージョンを [Passive] (パッシブ) に指定する必要があります。

7. [Edit routing status] (ルーティングステータスの編集) を選択します。
8. 表示されるダイアログボックスで、各リージョンの [Routing status] (ルーティングステータス) として [Active] (アクティブ) または [Passive] (パッシブ) を選択します。

   アクティブな状態では、トラフィックがリージョンにルーティングされます。パッシブな状態では、トラフィックがリージョンに誘導されなくなります。

マルチリージョンアクセスポイントのフェイルオーバー設定を作成する場合、またはフェイルオーバーを開始する場合は、マルチリージョンアクセスポイントで少なくとも 1 つの AWS リージョン を [Active] (アクティブ) に、1 つのリージョンを [Passive] (パッシブ) に指定する必要があります。

9. [Save routing status] (ルーティングステータスを保存) を選択します。トラフィックがリダイレクトされるまでに約 2 分かかります。

マルチリージョンアクセスポイントの AWS リージョン のルーティングステータスを送信すると、ルーティングステータスの変更を検証できます。これらの変更を検証するには、Amazon CloudWatch の https://console.aws.amazon.com/cloudwatch/ にアクセスして、アクティブリージョンとパッシブリージョン間の Amazon S3 データリクエストトラフィック (GET および PUT リクエストなど) のシフトをモニタリングします。フェイルオーバー中に既存の接続は終了しません。既存の接続は、成功または失敗のステータスに達するまで継続されます。

AWS CLI を使用する場合

> **Note**
>
> マルチリージョンアクセスポイント AWS CLI のルーティングコマンドは、次の 5 つのリージョンのいずれかに対して実行できます。
>
> - `ap-southeast-2`
> - `ap-northeast-1`
> - `us-east-1`
> - `us-west-2`
> - `eu-west-1`

次のコマンド例では、現在のマルチリージョンアクセスポイントのルート設定を更新します。バケットのアクティブまたはパッシブステータスを更新するには、`TrafficDialPercentage` の値をアクティブの場合は `100` に、パッシブの場合は `0` に設定します。この例では、*`DOC-EXAMPLE-BUCKET-1`* がアクティブに設定され、*`DOC-EXAMPLE-BUCKET-2`* がパッシブに設定されています。このコマンド例を実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control submit-multi-region-access-point-routes
```

```
--region ap-southeast-2
--account-id 111122223333
--mrap MultiRegionAccessPoint_ARN
--route-updates Bucket=DOC-EXAMPLE-BUCKET-1,TrafficDialPercentage=100
                Bucket=DOC-EXAMPLE-BUCKET-2,TrafficDialPercentage=0
```

次のコマンド例では、更新されたマルチリージョンアクセスポイントのルーティング設定を取得します。このコマンド例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
--mrap MultiRegionAccessPoint_ARN
```

フェイルオーバーの開始

マルチリージョンアクセスポイントの詳細ページの [Failover configuration] (フェイルオーバー設定) セクションで [Failover] (フェイルオーバー) を選択してフェイルオーバーを開始すると、Amazon S3 リクエストトラフィックは自動的に代替の AWS リージョン に移行されます。フェイルオーバープロセスは 2 分以内に完了します。

フェイルオーバーは、(マルチリージョンアクセスポイントがサポートされている 17 のリージョンのうち) 任意の 2 つの AWS リージョン で一度に開始できます。その後、フェイルオーバーイベントが AWS CloudTrail にログインされます。フェイルオーバーが完了すると、Amazon S3 トラフィックと、Amazon CloudWatch の新しいアクティブリージョンへのトラフィックルーティングの更新をモニタリングできます。

> **Important**
>
> データレプリケーション中にすべてのメタデータとオブジェクトをバケット間で同期させるには、フェイルオーバーコントロールを設定する前に、双方向のレプリケーションルールを作成し、レプリカ変更の同期を有効にすることをお勧めします。
> 双方向のレプリケーションルールにより、トラフィックがフェイルオーバーする Amazon S3 バケットにデータが書き込まれると、そのデータがソースバケットにレプリケートされます。レプリカ変更の同期により、双方向のレプリケーション中にオブジェクトメタデータもバケット間で確実に同期されます。
> フェイルオーバーをサポートするレプリケーションの設定については、「the section called “バケットレプリケーション”」を参照してください。

レプリケートされたバケット間のフェイルオーバーを開始するには

1. AWS マネジメントコンソールにサインインします。
2. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
3. 左のナビゲーションペインで、[Multi-Region Access Points] (マルチリージョンアクセスポイント) を選択します。
4. フェイルオーバーを開始するために使用する、マルチリージョンアクセスポイントを選択します。
5. [Replication and failover] (レプリケーションとフェイルオーバー) タブを選択します。
6. [Failover configuration] (フェイルオーバー設定) セクションまでスクロールして、AWS リージョン を 2 つ選択します。

   > **Note**
   >
   > フェイルオーバーを開始するには、マルチリージョンアクセスポイントで少なくとも 1 つの AWS リージョン を [Active] (アクティブ) に、1 つのリージョンを [Passive] (パッシブ) に指定する必要があります。アクティブな状態では、トラフィックがリージョンに誘導されます。パッシブな状態では、トラフィックがリージョンへ誘導されなくなります。

7. [フェイルオーバー] を選択します。
8. ダイアログボックスで、再度 [Failover] (フェイルオーバー) を選択して、フェイルオーバープロセスを開始します。このプロセス中に、2 つのリージョンのルーティングステータスが自動的に切り替わります。新しいトラフィックはすべてアクティブになったリージョンに誘導され、トラフィックはパッシブになったリージョンへ誘導されなくなります。トラフィックがリダイレクトされるまでに約 2 分かかります。

   フェイルオーバープロセスを開始すると、トラフィックの変化を検証できます。これらの変更を検証するには、Amazon CloudWatch の https://console.aws.amazon.com/cloudwatch/ にアクセスして、アクティブリージョンとパッシブリージョン間の Amazon S3 データリクエストトラフィック (GET および PUT リクエストなど) のシフトをモニタリングします。フェイルオーバー中に既存の接続は終了しません。既存の接続は、成功または失敗のステータスになるまで継続されます。

Amazon S3 マルチリージョンアクセスポイントアクセスポイントルーティングコントロールを表示する

S3 コンソールの使用

Amazon S3 マルチリージョンアクセスポイントのルーティングコントロールを表示するには

1. AWS マネジメントコンソールにサインインします。
2. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
3. 左のナビゲーションペインで、[Multi-Region Access Points] (マルチリージョンアクセスポイント) を選択します。
4. 確認するマルチリージョンアクセスポイントを選択します。
5. [Replication and failover] (レプリケーションとフェイルオーバー) タブを選択します。このページには、マルチリージョンアクセスポイントのルーティング設定の詳細と概要、関連するレプリケーションルール、およびレプリケーションメトリクスが表示されます。リージョンのルーティングステータスは、[Failover configuration] (フェイルオーバー設定) セクションで確認できます。

AWS CLI を使用する場合

次の例の AWS CLI コマンドでは、指定されたリージョンの、現在のマルチリージョンアクセスポイントのルート設定 を取得します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
--mrap MultiRegionAccessPoint_ARN
```

> **Note**
>
> このコマンドは、次の 5 つのリージョンに対してのみ実行できます。
>
> - ap-southeast-2
> - ap-northeast-1
> - us-east-1
> - us-west-2

- eu-west-1

Amazon S3 マルチリージョンアクセスポイントフェイルオーバーコントロールのエラー

マルチリージョンアクセスポイントのフェイルオーバー設定を更新すると、次のいずれかのエラーが発生する可能性があります。

- HTTP 400 Bad Request: このエラーは、フェイルオーバー設定の更新中に無効なマルチリージョンアクセスポイント ARN を入力した場合に発生する可能性があります。マルチリージョンアクセスポイント ARN は、マルチリージョンアクセスポイントポリシーを確認することで確認できます。マルチリージョンアクセスポイントポリシーを確認または更新するには、「マルチリージョンアクセスポイントポリシーの編集」を参照してください。このエラーは、Amazon S3 マルチリージョンアクセスポイントのフェイルオーバーコントロールの更新中に空の文字列またはランダムな文字列を使用した場合にも発生する可能性があります。必ずマルチリージョンアクセスポイント ARN 形式を使用してください。

  `arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`
- HTTP 503 Slow Down: このエラーは、短期間に送信するリクエストが多すぎる場合に発生します。拒否されたリクエストはエラーになります。
- HTTP 409 Conflict: このエラーは、2 つ以上の同時ルート設定更新リクエストが 1 つのマルチリージョンアクセスポイントを対象としている場合に発生します。最初のリクエストは成功しますが、他のリクエストはエラーで失敗します。
- HTTP 405 Method Not Allowed: このエラーは、フェイルオーバーの開始時に AWS リージョン が 1 つしかないマルチリージョンアクセスポイントを選択した場合に発生します。フェイルオーバーを開始する前に、2 つのリージョンを選択する必要があります。それ以外の場合は、エラーが返ります。

マルチリージョンアクセスポイントで使用するためのレプリケーションの設定

マルチリージョンアクセスポイントエンドポイントに対してリクエストを行うと、Amazon S3 は、最も近接したバケットにリクエストを自動的にルーティングします。この決定を行う際には、Amazon S3 ではリクエストの内容は考慮されません。オブジェクトの GET へのリクエストを行う場合、リクエストが、このオブジェクトのコピーを持たないバケットにルーティングされる可

能性もあります。その場合は、HTTP ステータスコード 404 (見つかりません) エラーが表示されます。マルチリージョンアクセスポイントのリクエストルーティングの詳細については、「the section called “リクエストルーティング”」を参照してください。

リクエストを受け取るバケットに関係なく、マルチリージョンアクセスポイントでオブジェクトを取得できるようにするには、Amazon S3 クロスリージョンレプリケーションを設定する必要があります。

例えば、3 つのバケットを持つマルチリージョンアクセスポイントについて考えてみましょう。

- オブジェクト `my-image.jpg` を含む、リージョン `us-west-2` の `my-bucket-usw2` という名前のバケット
- オブジェクト `my-image.jpg` を含む、リージョン `ap-south-1` の `my-bucket-aps1` という名前のバケット
- オブジェクト`my-image.jpg` を含まないリージョン `eu-central-1` の `my-bucket-euc1` という名前のバケット

このような状況では、オブジェクト `my-image.jpg` に対して `GetObject` リクエストを行った場合、そのリクエストの成功は、どのバケットがリクエストを受け取ったかによって異なります。Amazon S3 はリクエストの内容を考慮しないため、バケットが最も近接していると応答する場合、`GetObject` リクエストを `my-bucket-euc1` バケットにルーティングします。オブジェクトがマルチリージョンアクセスポイントのバケット内にある場合でも、リクエストを受け取った個々のバケットにオブジェクトが含まれていないため、404 Not Found エラーが発生します。

クロスリージョンレプリケーション (CRR) を有効にすると、この結果を回避できます。適切なレプリケーションルールを使用すると、`my-image.jpg` オブジェクトは `my-bucket-euc1` バケットにコピーされます。そのため、Amazon S3 がリクエストをそのバケットにルーティングすれば、オブジェクトを取得できるようになります。

レプリケーションは、マルチリージョンアクセスポイントに割り当てられているバケットでは通常どおりに機能します。Amazon S3 は、マルチリージョンアクセスポイントにあるバケットに対して特別なレプリケーション処理を実行しません。バケットでレプリケーションを設定する方法については、「ライブレプリケーションの設定」を参照してください。

マルチリージョンアクセスポイントでのレプリケーションの使用に関する推奨事項

マルチリージョンアクセスポイントでの操作時にレプリケーションパフォーマンスを最大限に高めたい場合は、次のことをお勧めします。

- S3 Replication Time Control (S3 RTC) を設定します。S3 RTC を使用して、予測可能な時間枠内で異なるリージョン間でデータをレプリケートできます。S3 RTC は、Simple Storage Service (Amazon S3) 内に保存されている新規オブジェクトの 99.99% を 15 分以内にレプリケートします (サービスレベルアグリーメントに基づく)。詳細については、「the section called “S3 Replication Time Control の有効化”」を参照してください。S3 RTC を使用するための追加料金はかかります。詳細については、「Amazon S3 の料金」を参照してください。
- また、双方向レプリケーションを有効にして、マルチリージョンアクセスポイントを介してバケットが更新されるときにバケットの同期を維持することをサポートします。詳細については、「the section called “マルチリージョンアクセスポイント用の双方向レプリケーションルールを作成する”」を参照してください。
- クロスアカウントのマルチリージョンアクセスポイントを作成して、AWS アカウント データを別のバケットに複製します。このアプローチでは、アカウントレベルで分離できるため、ソースバケット以外のさまざまなリージョンのさまざまなアカウントからデータにアクセスして、データを複製できます。クロスアカウントのマルチリージョンアクセスポイントの設定には、追加費用はかかりません。バケットの所有者であっても、マルチリージョンアクセスポイントを所有していない場合、データ転送とリクエストの費用のみがかかります。マルチリージョンアクセスポイントの所有者の場合は、データルーティングとインターネットアクセラレーションの費用がかかります。詳細については、「Amazon S3 の料金」を参照してください。
- 各レプリケーションルールでレプリカ変更同期を有効にすると、オブジェクトへのメタデータの変更も同期されます。詳細については、「レプリカ変更の同期の有効化する」を参照してください。
- Amazon CloudWatch メトリクスを有効にして、レプリケーションイベントをモニタリングします。CloudWatch メトリクス料金が適用されます。詳細については、「Amazon CloudWatch の料金」を参照してください。

トピック

- マルチリージョンアクセスポイント用の一方向レプリケーションルールを作成する
- マルチリージョンアクセスポイント用の双方向レプリケーションルールを作成する
- マルチリージョンアクセスポイント用のレプリケーションルールを表示する

マルチリージョンアクセスポイント用の一方向レプリケーションルールを作成する

レプリケーションルールでは、異なるバケット間でオブジェクトを自動的に非同期コピーできます。一方向レプリケーションルールを使用すると、1 つの AWS リージョン のソースバケットから別のリージョンの送信先バケットにデータを完全に、確実にレプリケートできます。一方向レプリケーションを設定する場合は、レプリケート元バケット (DOC-EXAMPLE-BUCKET-1) からレプリケート

先バケット (DOC-EXAMPLE-BUCKET-2) へのレプリケーションルールを作成します。これらの一方向レプリケーションルールは、すべてのレプリケーションルールと同様に、Amazon S3 バケット全体に適用することも、オブジェクトのサブセットをプレフィックスまたはオブジェクトタグでフィルタリングして適用することも可能です。

> **Important**
>
> ユーザーがレプリケート先バケット内のオブジェクトのみを使用する場合は、一方向レプリケーションを使用することをお勧めします。ユーザーがレプリケート先バケットのオブジェクトをアップロードまたは変更する場合は、双方向レプリケーションを使用してすべてのバケットを同期させてください。また、マルチリージョンアクセスポイントをフェイルオーバーに使用する場合は、双方向レプリケーションをお勧めします。双方向レプリケーションを設定するには、「the section called “マルチリージョンアクセスポイント用の双方向レプリケーションルールを作成する”」を参照してください。

マルチリージョンアクセスポイント用の一方向レプリケーションルールを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Multi-Region Access Points] (マルチリージョンアクセスポイント) を選択します。
3. マルチリージョンアクセスポイント名を選択します。
4. [Replication and failove] (レプリケーションとフェイルオーバー) タブを選択します。
5. [Replication rules] (レプリケーションルール) セクションまでスクロールして、[Create replication rules] (レプリケーションルールの作成) を選択します。レプリケーションルールを作成するための十分なアクセス権限があることを確認してください。権限がないと、バージョニングが無効になります。

   > **Note**
   >
   > レプリケーションルールは、自分が所有するアカウントのバケットに対してのみ作成できます。外部バケットのレプリケーションルールを作成するには、バケットの所有者がそれらのバケットのレプリケーションルールを作成する必要があります。

6. [レプリケーションルールの作成] ページで、[1 つ以上のレプリケート元バケットから 1 つ以上のレプリケート先バケットにオブジェクトをレプリケート] テンプレートを選択します。

> **Important**
>
> このテンプレートを使用してレプリケーションルールを作成すると、バケットに既に割り当てられている既存のレプリケーションルールが置き換えられます。
> 既存のレプリケーションルールを置き換える代わりに追加または変更するには、コンソールで各バケットの [Management] (管理) タブに移動し、[Replication rules] (レプリケーションルール) セクションでルールを編集します。AWS CLI (SDK)、または REST API を使用して、既存のレプリケーションルールを追加または変更することもできます。詳細については、「レプリケーション設定」を参照してください。

7. [送信元と送信先] セクションの [レプリケート元バケット] で、オブジェクトのレプリケート元となるバケットを 1 つ以上選択します。レプリケーション用に選択したすべてのバケット (レプリケート元とレプリケート先) では S3 バージョニングが有効になっている必要があり、各バケットは異なる AWS リージョン に配置されている必要があります。オブジェクトのバージョニングを管理する方法の詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

   [レプリケート先バケット] セクションで、オブジェクトのレプリケート先となるバケットを 1 つ以上選択します。

   > **Note**
   >
   > レプリケーションを確立するために必要な読み取り権限とレプリケート権限があることを確認してください。権限がないと、エラーが発生します。詳細については、「IAM ロールの作成」を参照してください。

8. [Replication rule configuration] (レプリケーションルール設定) セクションで、作成時にレプリケーションルールを[Enabled] (有効) にするか [Disabled] (無効) にするかを選択します。

   > **Note**
   >
   > [Replication rule name] (レプリケーションルール名) ボックスに名前を入力することはできません。レプリケーションルール名は、レプリケーションルールを作成したときの構成に基づいて生成されます。

9. [Scope] (スコープ) セクションで、レプリケーションに適したスコープを選択します。

- バケット全体をレプリケートするには、[Apply to all objects in the bucket] (バケット内のすべてのオブジェクトに適用) を選択します。
- バケットのオブジェクトのサブセットをレプリケートするには、[Limit the scope of this rule using one or more filters] (1 つまたは複数のフィルターを使用してこのルールの適用範囲を制限します) を選択します。

  プレフィックス、オブジェクトタグ、またはその両方の組み合わせでフィルターできます。

  - 同じ文字列で始まる名前 (`pictures` など) をもつすべてのオブジェクトにレプリケーションを制限するには、[Prefix] (プレフィクス) ボックスにプレフィックスを入力します。

    フォルダの名前をプレフィックスとして入力する場合は、`/` (スラッシュ) などの区切り文字を使用して、階層のレベルを示す必要があります (例:`pictures/`)。プレフィックスの詳細については、「プレフィックスを使用してオブジェクトを整理する」を参照してください。
  - 1 つまたは複数のオブジェクトタグを持つすべてのオブジェクトをレプリケートするには、[Add tag] (タグを追加) を選択し、ボックスにキーと値のペアを入力します。別のタグを追加するには、この手順を繰り返します。オブジェクトロックの詳細については、タグを使用してストレージを分類するを参照してください。

10. [Additional replication options] (その他のレプリケーションオプション) セクションまでスクロールして、適用するレプリケーションオプションを選択します。

> **Note**
>
> 以下のオプションの適用をお勧めします。
>
> - S3 Replication Time Control (S3 RTC) – 予測可能な時間枠内で異なるリージョン間でデータをレプリケートするには、S3 Replication Time Control (S3 RTC) を使用します。S3 RTC は、Simple Storage Service (Amazon S3) 内に保存されている新規オブジェクトの 99.99% を 15 分以内にレプリケートします (サービスレベルアグリーメントに基づく)。詳細については、「the section called “S3 Replication Time Control の有効化”」を参照してください。
> - レプリケーションメトリクスと通知 – Amazon CloudWatch メトリクスを有効にして、レプリケーションイベントをモニタリングできます。
> - 削除マーカーのレプリケーション — S3 の削除オペレーションによって作成された削除マーカーがレプリケートされます。ライフサイクルルールによって作成された削除マーカーはレプリケートされません。詳細については、「バケット間での削除マーカーのレプリケーション」を参照してください。

S3 RTC と CloudWatch のレプリケーションメトリクスと通知には追加料金がかかります。詳細については、「Amazon S3 の料金」および「Amazon CloudWatch の料金」を参照してください。

11. 既存のレプリケーションルールを置き換える新しいレプリケーションルールを作成する場合は、[Create replication rules] (レプリケーションルールを作成) を選択すると、既存のレプリケーションルールが上書きされることに同意します。
12. [レプリケーションルールの作成] を選択し、新しい一方向レプリケーションルールを作成して保存します。

マルチリージョンアクセスポイント用の双方向レプリケーションルールを作成する

レプリケーションルールでは、異なるバケット間でオブジェクトを自動的に非同期コピーできます。双方向レプリケーションルールは、異なる AWS リージョン にあるデータが複数のバケット間で完全に同期されることを保証します。双方向レプリケーションを設定する場合、レプリケート元バケット (DOC-EXAMPLE-BUCKET-1) から、レプリカが含まれているバケット (DOC-EXAMPLE-BUCKET-2) に対するレプリケーションルールを作成します。次に、レプリカを含むバケット (DOC-EXAMPLE-BUCKET-2) からレプリケート元バケット (DOC-EXAMPLE-BUCKET-1) に対する 2 つ目のレプリケーションルールを作成します。

これらの双方向レプリケーションルールは、すべてのレプリケーションルールと同様に、Amazon S3 バケット全体に適用することも、オブジェクトのサブセットをプレフィックスまたはオブジェクトタグでフィルタリングして適用することも可能です。各レプリケーションルールでレプリカ変更同期を有効にすると、オブジェクトへのメタデータの変更も同期されます。Amazon S3 コンソール、AWS CLI、AWS SDK、Amazon S3 REST API、または AWS CloudFormation を使用して、レプリカ変更の同期を有効にすることができます。

Amazon CloudWatch のオブジェクトとオブジェクトメタデータのレプリケーションの進行状況をモニタリングするには、S3 レプリケーションメトリクスと通知を有効にします。詳細については、「レプリケーションメトリクスと Amazon S3 イベント通知による、進捗状況のモニタリング」を参照してください。

マルチリージョンアクセスポイント用の双方向レプリケーションルールを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。

2. ナビゲーションペインで、[Multi-Region Access Points] (マルチリージョンアクセスポイント) を選択します。
3. 更新するマルチリージョンアクセスポイントの名前を選択します。
4. [Replication and failove] (レプリケーションとフェイルオーバー) タブを選択します。
5. [Replication rules] (レプリケーションルール) セクションまでスクロールして、[Create replication rules] (レプリケーションルールの作成) を選択します。
6. [Create replication rules] (レプリケーションルールの作成) ページで、[Replicate objects among all specified buckets] (指定されたすべてのバケット間でオブジェクトをレプリケート) テンプレートを選択します。[Replicate objects among all specified buckets] (指定されたすべてのバケット間でオブジェクトをレプリケート) テンプレートは、バケットの双方向レプリケーション (フェイルオーバー機能付き) を設定します。

> **Important**
>
> このテンプレートを使用してレプリケーションルールを作成すると、バケットに既に割り当てられている既存のレプリケーションルールが置き換えられます。
> 既存のレプリケーションルールを置き換える代わりに追加または変更するには、コンソールで各バケットの [Management] (管理) タブに移動し、[Replication rules] (レプリケーションルール) セクションでルールを編集します。AWS CLI、AWS SDK、または Amazon S3 REST API を使用して既存のレプリケーションルールを追加または変更することもできます。詳細については、「レプリケーション設定」を参照してください。

7. [Buckets] (バケット) セクションで、オブジェクトの複製元となるバケットを少なくとも 2 つ選択します。レプリケーション用に選択したすべてのバケットで S3 バージョニングが有効になっている必要があり、各バケットは異なる AWS リージョン にある必要があります。オブジェクトのバージョニングを管理する方法の詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

> **Note**
>
> レプリケーションを確立するために必要な読み取り権限とレプリケート権限があることを確認してください。権限がないと、エラーが発生します。詳細については、「IAM ロールの作成」を参照してください。

8. [Replication rule configuration] (レプリケーションルール設定) セクションで、作成時にレプリケーションルールを[Enabled] (有効) にするか [Disabled] (無効) にするかを選択します。

> **Note**
>
> [Replication rule name] (レプリケーションルール名) ボックスに名前を入力することはできません。レプリケーションルール名は、レプリケーションルールを作成したときの構成に基づいて生成されます。

9. [Scope] (スコープ) セクションで、レプリケーションに適したスコープを選択します。

   - バケット全体をレプリケートするには、[Apply to all objects in the bucket] (バケット内のすべてのオブジェクトに適用) を選択します。
   - バケットのオブジェクトのサブセットをレプリケートするには、[Limit the scope of this rule using one or more filters] (1 つまたは複数のフィルターを使用してこのルールの適用範囲を制限します) を選択します。

     プレフィックス、オブジェクトタグ、またはその両方の組み合わせでフィルターできます。

     - 同じ文字列で始まる名前 (`pictures` など) をもつすべてのオブジェクトにレプリケーションを制限するには、[Prefix] (プレフィクス) ボックスにプレフィックスを入力します。

       フォルダの名前をプレフィックスとして入力する場合は、最後の文字に / (スラッシュ) を使用する必要があります (例: `pictures/`)。
     - 1 つまたは複数のオブジェクトタグを持つすべてのオブジェクトをレプリケートするには、[Add tag] (タグを追加) を選択し、ボックスにキーと値のペアを入力します。別のタグを追加するには、この手順を繰り返します。オブジェクトロックの詳細については、タグを使用してストレージを分類するを参照してください。

10. [Additional replication options] (その他のレプリケーションオプション) セクションまでスクロールして、適用するレプリケーションオプションを選択します。

> **Note**
>
> 特にマルチリージョンアクセスポイントをフェイルオーバーをサポートするように設定する場合は、次のオプションを適用することをお勧めします。
>
> - S3 Replication Time Control (S3 RTC) – 予測可能な時間枠内で異なるリージョン間でデータをレプリケートするには、S3 Replication Time Control (S3 RTC) を使用します。S3 RTC は、Simple Storage Service (Amazon S3) 内に保存されている新規オブジェクトの 99.99% を 15 分以内にレプリケートします (サービスレベルアグリーメン

トに基づく)。詳細については、「the section called “S3 Replication Time Control の有効化”」を参照してください。

- レプリケーションメトリクスと通知 – Amazon CloudWatch メトリクスを有効にして、レプリケーションイベントをモニタリングできます。
- 削除マーカーのレプリケーション — S3 の削除オペレーションによって作成された削除マーカーがレプリケートされます。ライフサイクルルールによって作成された削除マーカーはレプリケートされません。詳細については、「バケット間での削除マーカーのレプリケーション」を参照してください。
- レプリカ変更同期 – 各レプリケーションルールでレプリカ変更同期を有効にすると、オブジェクトへのメタデータの変更も同期されます。詳細については、「レプリカ変更の同期の有効化する」を参照してください。

S3 RTC と CloudWatch のレプリケーションメトリクスと通知には追加料金がかかります。詳細については、「Amazon S3 の料金」および「Amazon CloudWatch の料金」を参照してください。

11. 既存のレプリケーションルールを置き換える新しいレプリケーションルールを作成する場合は、[Create replication rules] (レプリケーションルールを作成) を選択すると、既存のレプリケーションルールが上書きされることに同意します。
12. [Create replication rules] (レプリケーションルールを作成) を選択し、新しい双方向レプリケーションルールを作成して保存します。

マルチリージョンアクセスポイント用のレプリケーションルールを表示する

マルチリージョンアクセスポイントでは、一方向レプリケーションルールまたは双方向レプリケーションルールを設定できます。レプリケーションルールの管理方法の詳細については、「Amazon S3 コンソールを使用したレプリケーションルールの管理」を参照してください。

マルチリージョンアクセスポイント用のレプリケーションルールを表示するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Multi-Region Access Points] (マルチリージョンアクセスポイント) を選択します。
3. マルチリージョンアクセスポイント名を選択します。

4. [Replication and failove] (レプリケーションとフェイルオーバー) タブを選択します。
5. [レプリケーションルール] セクションまで、下にスクロールします。このセクションには、マルチリージョンアクセスポイント用に作成されたすべてのレプリケーションルールが一覧表示されます。

> **Note**
>
> 別のアカウントから、このマルチリージョンアクセスポイントにバケットを追加した場合、そのバケットのレプリケーションルールを表示するにはバケット所有者からの `s3:GetBucketReplication` 権限が必要です。

マルチリージョンアクセスポイントでのサポートされている API オペレーションの使用

Amazon S3 は、マルチリージョンアクセスポイントを管理するための一連オペレーションを提供します。Amazon S3 は、これらのオペレーションの一部を同期的に処理し、一部を非同期的に処理します。非同期オペレーションを呼び出すと、Amazon S3 は最初にリクエストされたオペレーションを同期的に承認します。承認が成功すると、リクエストされたオペレーションの進行状況と結果を追跡するために使用できるトークンが Amazon S3 から返されます。

> **Note**
>
> Amazon S3 コンソールを介して実行されるリクエストは常に同期されます。コンソールは、リクエストが完了するまで待機してから、別のリクエストを送信できるようにします。

コンソールを使用するか、AWS CLI、AWS SDK、または REST API で `DescribeMultiRegionAccessPointOperation` を使用して、非同期オペレーションの現在のステータスと結果を表示できます。Amazon S3 は、非同期オペレーションへの応答で追跡用トークンを提供します。その追跡用トークンを、`DescribeMultiRegionAccessPointOperation` への引数として含めます。追跡用トークンを含めると、Amazon S3 はその後、エラーや関連するリソース情報を含む、指定されたオペレーションの現在のステータスと結果を返します。Amazon S3 は `DescribeMultiRegionAccessPointOperation` オペレーションを同期的に実行します。

マルチリージョンアクセスポイントの作成または保守のすべてのコントロールプレーンリクエストは、US West (Oregon) リージョンにルーティングされる必要があります。マルチリージョンアク

セスポイントのデータプレーンリクエストでは、リージョンを指定する必要はありません。マルチリージョンアクセスポイントのフェイルオーバーコントロールプレーンの場合、リクエストは 5 つのサポートされているリージョンのいずれかにルーティングされる必要があります。マルチリージョンアクセスポイントのサポートされているリージョンの詳細については、「マルチリージョンアクセスポイントの制約および制限事項」を参照してください。

さらに、`s3:ListAllMyBuckets` アクセス許可を、ユーザー、ロール、またはマルチリージョンアクセスポイントを管理するためのリクエストを作成する他の AWS Identity and Access Management (IAM) エンティティに付与します。

以下の例は、Amazon S3 の互換性のあるオペレーションでマルチリージョンアクセスポイントを使用する方法を示しています。

トピック

- マルチリージョンアクセスポイントと AWS のサービス および AWS SDK との互換性
- S3 オペレーションとマルチリージョンアクセスポイントの互換性
- マルチリージョンアクセスポイントのルーティング設定を表示する
- 基になる Amazon S3 バケットポリシーを更新する
- マルチリージョンアクセスポイントのルート設定を更新する
- マルチリージョンアクセスポイントのバケットへのオブジェクトの追加
- マルチリージョンアクセスポイントからのオブジェクトの取得
- マルチリージョンアクセスポイントの基礎となるバケットに保存されているオブジェクトのリスト表示
- マルチリージョンアクセスポイントで署名付き URL を使用する
- マルチリージョンアクセスポイントでリクエスタ支払いが設定されたバケットを使用する

マルチリージョンアクセスポイントと AWS のサービス および AWS SDK との互換性

Amazon S3 バケット名を必要とするアプリケーションで、マルチリージョンアクセスポイントを使用するには、AWS SDK を使用してリクエストするときにマルチリージョンアクセスポイントの Amazon リソースネーム (ARN) を使用します。AWS SDK がマルチリージョンのアクセスポイントと互換性があるかどうかを確認するには、「AWS SDK との互換性」を参照してください。

S3 オペレーションとマルチリージョンアクセスポイントの互換性

次の Amazon S3 データプレーン API オペレーションを使用して、マルチリージョンアクセスポイントに関連付けられているバケット内のオブジェクトに対してアクションを実行できます。以下の S3 オペレーションでは、マルチリージョンアクセスポイント ARN を使用できます。

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateMultipartUpload
- DeleteObject
- DeleteObjectTagging
- GetObject
- GetObjectAcl
- GetObjectLegalHold
- GetObjectRetention
- GetObjectTagging
- HeadObject
- ListMultipartUploads
- ListObjectsV2
- ListParts
- PutObject
- PutObjectAcl
- PutObjectLegalHold
- PutObjectRetention
- PutObjectTagging
- RestoreObject
- UploadPart

> **Note**
>
> マルチリージョンアクセスポイントは、マルチリージョンアクセスポイント ARN を使用する場合に限り、マルチリージョンアクセスポイントを宛先とするコピー操作をサポートします。

マルチリージョンアクセスポイントの作成と管理には、次の Amazon S3 コントロールプレーンオペレーションを使用できます。

- CreateMultiRegionAccessPoint
- DescribeMultiRegionAccessPointOperation
- GetMultiRegionAccessPoint
- GetMultiRegionAccessPointPolicy
- GetMultiRegionAccessPointPolicyStatus
- GetMultiRegionAccessPointRoutes
- ListMultiRegionAccessPoints
- PutMultiRegionAccessPointPolicy
- SubmitMultiRegionAccessPointRoutes

マルチリージョンアクセスポイントのルーティング設定を表示する

AWS CLI

次のコマンド例では、マルチリージョンアクセスポイントのルート設定を取得して、バケットの現在のルーティングステータスを確認できるようにします。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
--mrap arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap
```

SDK for Java

次の SDK for Java コードは、マルチリージョンアクセスポイントのルート設定を取得して、バケットの現在のルーティングステータスを確認できるようにします。この構文例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
S3ControlClient s3ControlClient = S3ControlClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(credentialsProvider)
    .build();

GetMultiRegionAccessPointRoutesRequest request =
 GetMultiRegionAccessPointRoutesRequest.builder()
    .accountId("111122223333")
    .mrap("arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap")
    .build();

GetMultiRegionAccessPointRoutesResponse response =
 s3ControlClient.getMultiRegionAccessPointRoutes(request);
```

SDK for JavaScript

次の SDK for JavaScript コードは、マルチリージョンアクセスポイントのルート設定を取得して、バケットの現在のルーティングステータスを確認できるようにします。この構文例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
const REGION = 'us-east-1'

const s3ControlClient = new S3ControlClient({
  region: REGION
})

export const run = async () => {
  try {
    const data = await s3ControlClient.send(
      new GetMultiRegionAccessPointRoutesCommand({
        AccountId: '111122223333',
        Mrap: 'arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap',
      })
    )
    console.log('Success', data)
    return data
  } catch (err) {
```

```
    console.log('Error', err)
  }
}

run()
```

SDK for Python

次の SDK for Python コードは、マルチリージョンアクセスポイントのルート設定を取得して、バケットの現在のルーティングステータスを確認できるようにします。この構文例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
s3.get_multi_region_access_point_routes(
        AccountId=111122223333,
        Mrap=arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap)['Routes']
```

基になる Amazon S3 バケットポリシーを更新する

適切なアクセスを許可するには、基になる Amazon S3 バケットポリシーも更新する必要があります。以下の例では、アクセスコントロールをマルチリージョンアクセスポイントポリシーに委任します。マルチリージョンアクセスポイントポリシーにアクセスコントロールを委任すると、マルチリージョンアクセスポイントを介してリクエストを行うときバケットポリシーがアクセスコントロールに使用されなくなります。

マルチリージョンアクセスポイントポリシーにアクセスコントロールを委任するバケットポリシーの例を次に示します。このバケットポリシーの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。このポリシーを AWS CLI `put-bucket-policy` コマンドを使用して適用するには、次の例に示すように、ポリシーをファイルに保存します (例：`policy.json`)。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": { "AWS": "*" },
    "Effect": "Allow",
    "Action": ["s3:*"],
    "Resource": ["arn:aws:s3:::111122223333/*", "arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
    "Condition": {
      "StringEquals": {
        "s3:DataAccessPointAccount": "444455556666"
```

```
      }
    }
  }
}
```

次の put-bucket-policy 例では、更新された S3 バケットポリシーを S3 バケットに関連付けます。

```
aws s3api put-bucket-policy
  --bucket DOC-EXAMPLE-BUCKET
  --policy file:///tmp/policy.json
```

マルチリージョンアクセスポイントのルート設定を更新する

以下のコマンド例では、マルチリージョンアクセスポイントのルート設定も更新します。マルチリージョンアクセスポイントのルートコマンドは、次の 5 つのリージョンに対して実行できます。

- ap-southeast-2
- ap-northeast-1
- us-east-1
- us-west-2
- eu-west-1

マルチリージョンアクセスポイントのルーティング設定では、バケットをアクティブまたはパッシブのルーティングステータスに設定できます。アクティブバケットはトラフィックを受信しますが、パッシブバケットは受信しません。バケットの TrafficDialPercentage 値を 100 に設定するとアクティブ、0 に設定するとパッシブになり、バケットのルーティングステータスを設定できます。

AWS CLI

以下のコマンド例では、マルチリージョンアクセスポイントのルーティング設定も更新します。この例では、*DOC-EXAMPLE-BUCKET1* がアクティブステータスに設定され、*DOC-EXAMPLE-BUCKET2* がパッシブに設定されています。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control submit-multi-region-access-point-routes
--region ap-southeast-2
--account-id 111122223333
```

```
--mrap arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap
--route-updates Bucket=DOC-EXAMPLE-BUCKET1,TrafficDialPercentage=100
                Bucket=DOC-EXAMPLE-BUCKET2,TrafficDialPercentage=0
```

SDK for Java

次の SDK for Java コードは、マルチリージョンアクセスポイントのルーティング設定も更新します。この構文例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
S3ControlClient s3ControlClient = S3ControlClient.builder()
    .region(Region.ap-southeast-2)
    .credentialsProvider(credentialsProvider)
    .build();

SubmitMultiRegionAccessPointRoutesRequest request =
 SubmitMultiRegionAccessPointRoutesRequest.builder()
    .accountId("111122223333")
    .mrap("arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap")
    .routeUpdates(
        MultiRegionAccessPointRoute.builder()
            .region("eu-west-1")
            .trafficDialPercentage(100)
            .build(),
        MultiRegionAccessPointRoute.builder()
            .region("ca-central-1")
            .bucket("111122223333")
            .trafficDialPercentage(0)
            .build()
    )
    .build();

SubmitMultiRegionAccessPointRoutesResponse response =
 s3ControlClient.submitMultiRegionAccessPointRoutes(request);
```

SDK for JavaScript

次の SDK for JavaScript コードは、マルチリージョンアクセスポイントのルート設定も更新します。この構文例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
const REGION = 'ap-southeast-2'
```

```
const s3ControlClient = new S3ControlClient({
  region: REGION
})

export const run = async () => {
  try {
    const data = await s3ControlClient.send(
      new SubmitMultiRegionAccessPointRoutesCommand({
        AccountId: '111122223333',
        Mrap: 'arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap',
        RouteUpdates: [
          {
            Region: 'eu-west-1',
            TrafficDialPercentage: 100,
          },
          {
            Region: 'ca-central-1',
            Bucket: 'DOC-EXAMPLE-BUCKET1',
            TrafficDialPercentage: 0,
          },
        ],
      })
    )
    console.log('Success', data)
    return data
  } catch (err) {
    console.log('Error', err)
  }
}

run()
```

SDK for Python

次の SDK for Python コードは、マルチリージョンアクセスポイントのルート設定も更新します。この構文例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
s3.submit_multi_region_access_point_routes(
        AccountId=111122223333,
        Mrap=arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap,
        RouteUpdates= [{
```

```
            'Bucket': DOC-EXAMPLE-BUCKET,
            'Region': ap-southeast-2,
            'TrafficDialPercentage': 10
        }])
```

マルチリージョンアクセスポイントのバケットへのオブジェクトの追加

マルチリージョンアクセスポイントに関連付けられているバケットにオブジェクトを追加するには、PutObject オペレーションを使用できます。マルチリージョンアクセスポイントのすべてのバケットを同期させるには、[クロスリージョンレプリケーション] を有効にします。

> **Note**
>
> このオペレーションを使用するには、マルチリージョンアクセスポイントの s3:PutObject アクセス許可が必要です。マルチリージョンアクセスポイントのアクセス許可の要件の詳細については、「許可」を参照してください。

AWS CLI

次の例のデータプレーンリクエストは、指定されたマルチリージョンアクセスポイントに *example.txt* をアップロードします。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3api put-object --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap --key example.txt --
body example.txt
```

SDK for Java

```
S3Client s3Client = S3Client.builder()
        .build();

PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
        .key("example.txt")
        .build();

s3Client.putObject(objectRequest, RequestBody.fromString("Hello S3!"));
```

SDK for JavaScript

```
const client = new S3Client({});

async function putObjectExample() {
    const command = new PutObjectCommand({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
        Key: "example.txt",
        Body: "Hello S3!",
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.put_object(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap',
    Key='example.txt',
    Body='Hello S3!'
)
```

マルチリージョンアクセスポイントからのオブジェクトの取得

マルチリージョンアクセスポイントからオブジェクトを取得するには、GetObject オペレーションを使用できます。

> **Note**
>
> この API オペレーションを使用するには、マルチリージョンアクセスポイントの s3:GetObject アクセス許可が必要です。マルチリージョンアクセスポイントのアクセス許可の要件の詳細については、「許可」を参照してください。

AWS CLI

次のデータプレーンリクエストの例では、指定されたマルチリージョンアクセスポイントから *example.txt* を取得し、それを *downloaded_example.txt* としてダウンロードします。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3api get-object --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap --
key example.txt downloaded_example.txt
```

SDK for Java

```
S3Client s3 = S3Client
   .builder()
   .build();

GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
    .key("example.txt")
    .build();

s3Client.getObject(getObjectRequest);
```

SDK for JavaScript

```
const client = new S3Client({})

async function getObjectExample() {
    const command = new GetObjectCommand({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
        Key: "example.txt"
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.get_object(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap',
    Key='example.txt'
)
```

マルチリージョンアクセスポイントの基礎となるバケットに保存されているオブジェクトのリスト表示

マルチリージョンアクセスポイントの基礎となるバケットに保存されたオブジェクトのリストを返すには、ListObjectsV2 オペレーションを使用します。次のコマンド例では、マルチリージョンアクセスポイントの ARN を使用して、マルチリージョンアクセスポイントのすべてのオブジェクトがリスト表示されます。この場合、マルチリージョンアクセスポイントの ARN は次のようになります。

`arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap`

> **Note**
>
> この API オペレーションを使用するには、マルチリージョンアクセスポイントと基礎となるバケットの `s3:ListBucket` アクセス許可が必要です。マルチリージョンアクセスポイントのアクセス許可の要件の詳細については、「許可」を参照してください。

AWS CLI

次のデータプレーンリクエストの例では、ARN で指定されたマルチリージョンアクセスポイントの基礎となるバケット内のオブジェクトをリスト表示します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3api list-objects-v2 --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap
```

SDK for Java

```
S3Client s3Client = S3Client.builder()
        .build();
```

```
String bucketName = "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap";

ListObjectsV2Request listObjectsRequest = ListObjectsV2Request
    .builder()
    .bucket(bucketName)
    .build();

 s3Client.listObjectsV2(listObjectsRequest);
```

SDK for JavaScript

```
const client = new S3Client({});

async function listObjectsExample() {
    const command = new ListObjectsV2Command({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.list_objects_v2(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap'
)
```

マルチリージョンアクセスポイントで署名付き URL を使用する

署名付き URL を使用すると、他のユーザーが Amazon S3 マルチリージョンアクセスポイントを介して Amazon S3 バケットにアクセスするための URL を生成することができます。署名付き URL を作成したら、その署名付き URL を S3 アップロード (PutObject) や S3 ダウンロード

(GetObject) などの特定のオブジェクトアクションに関連付けます。署名付き URL を共有すると、URL へのアクセス権を持つユーザーは、URL に埋め込まれたアクションを、元の署名ユーザーであるかのように実行できます。

署名付き URL には有効期限があります。有効期限に達すると、URL は機能しなくなります。

署名付き URL で S3 マルチリージョンアクセスポイントを使用する前に、Sigv4A アルゴリズムと AWS SDK との互換性を確認してください。ご使用の SDK バージョンが、グローバル AWS リージョン リクエストの署名に使用される署名実装として SigV4AA をサポートしていることを確認してください。Amazon S3 での署名付き URL の使用について詳しくは、「Sharing objects by using presigned URL」(署名付き URL を使用してオブジェクトを共有する) を参照してください。

以下の例は、署名付き URL と共にマルチリージョンアクセスポイントを使用する方法を示しています。これらの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

AWS CLI

```
aws s3 presign
 arn:aws:s3::123456789012:accesspoint/MultiRegionAccessPoint_alias/example-file.txt
```

SDK for Python

```
import logging
import boto3
from botocore.exceptions import ClientError

s3_client = boto3.client('s3',aws_access_key_id='xxx',aws_secret_access_key='xxx')
s3_client.generate_presigned_url(HttpMethod='PUT',ClientMethod="put_object",
 Params={'Bucket':'arn:aws:s3::123456789012:accesspoint/
abcdef0123456.mrap','Key':'example-file'})
```

SDK for Java

```
S3Presigner s3Presigner = S3Presigner.builder()
    .credentialsProvider(StsAssumeRoleCredentialsProvider.builder()
        .refreshRequest(assumeRole)
        .stsClient(stsClient)
        .build())
    .build();
```

```
GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
    .key("example-file")
    .build();

GetObjectPresignRequest preSignedReq = GetObjectPresignRequest.builder()
    .getObjectRequest(getObjectRequest)
    .signatureDuration(Duration.ofMinutes(10))
    .build();

PresignedGetObjectRequest presignedGetObjectRequest =
 s3Presigner.presignGetObject(preSignedReq);
```

> **Note**
>
> SigV4A を一時的なセキュリティ認証情報と共に使用する場合、例えば IAM ロールを使用する場合には、必ずグローバルエンドポイントではなく、AWS Security Token Service (AWS STS) 内のリージョンエンドポイントから一時的認証情報をリクエストしてください。AWS STS (`sts.amazonaws.com`) にグローバルエンドポイントを使用すると、AWS STS はグローバルエンドポイントから、Sig4A ではサポートされていない一時的な認証情報を生成します。その結果、エラーが発生します。この問題を解決するには、AWS STS 用のリージョンエンドポイントにリストされているいずれかのエンドポイントを使用してください。

マルチリージョンアクセスポイントでリクエスタ支払いが設定されたバケットを使用する

マルチリージョンアクセスポイントに関連付けられている S3 バケットがリクエスタ支払いを使用するように設定されている場合、リクエスタはバケットリクエスト、ダウンロードおよびマルチリージョンアクセスポイントに関連する費用を支払います。詳細については、「Amazon S3 の料金」を参照してください。

リクエスタ支払いバケットに接続されたマルチリージョンアクセスポイントへのデータプレーンリクエストの例を次に示します。

AWS CLI

リクエスタ支払いバケットに接続したマルチリージョンアクセスポイントからオブジェクトをダウンロードするには、get-object リクエストの一部として `--request-payer requester` を指

定する必要があります。また、バケット内のファイルの名前と、ダウンロードファイルを保存する場所を指定する必要があります。

```
aws s3api get-object --bucket MultiRegionAccessPoint_ARN --request-payer requester
 --key example-file-in-bucket.txt example-location-of-downloaded-file.txt
```

SDK for Java

リクエスタ支払いバケットに接続したマルチリージョンアクセスポイントからオブジェクトをダウンロードするには、GetObject リクエストの一部として RequestPayer.REQUESTER を指定する必要があります。また、バケット内のファイルの名前と、ファイルを保存する場所を指定する必要があります。

```
GetObjectResponse getObjectResponse = s3Client.getObject(GetObjectRequest.builder()
    .key("example-file.txt")
    .bucket("arn:aws:s3::
123456789012:accesspoint/abcdef0123456.mrap")
    .requestPayer(RequestPayer.REQUESTER)
    .build()
).response();
```

マルチリージョンアクセスポイントから基盤となるリソースへのリクエストのモニタリングと口グ記録

Amazon S3 は、マルチリージョンアクセスポイントを介して行われたリクエストと、アクセスポイントを管理する API オペレーションに対するリクエスト (CreateMultiRegionAccessPoint や GetMultiRegionAccessPointPolicy など) を口グに記録します。マルチリージョンアクセスポイントを介して Amazon S3 に行われたリクエストは、アクセスポイントのホスト名と共に、Amazon S3 サーバーのアクセス口グと AWS CloudTrail の口グに表示されます。アクセスポイントのホスト名は、*MRAP_alias*.accesspoint.s3-global.amazonaws.com という形式になります。例えば、次のバケットとマルチリージョンアクセスポイントの設定があるとします。

- オブジェクト my-image.jpg を含む、リージョン us-west-2 の my-bucket-usw2 という名前のバケット。
- オブジェクト my-image.jpg を含む、リージョン ap-south-1 の my-bucket-aps1 という名前のバケット。
- my-image.jpg という名前のオブジェクト を含まないリージョン eu-central-1 の my-bucket-euc1 という名前のバケット。

- 名前が my-mrap であり、エイリアスが mfzwi23gnjvgw.mrap であるマルチリージョンアクセスポイントは、3 つのバケットすべてからのリクエストを満たすように設定されています。
- お客様の AWS アカウント ID は 123456789012 です。

バケットから my-image.jpg を直接取得するリクエストは、ホスト名 *bucket_name*.s3.*Region*.amazonaws.com と共にログに表示されます。

代わりにマルチリージョンアクセスポイント経由でリクエストを行うと、Amazon S3 は最初に、異なるリージョン内のバケットのうち、どれが最も近いかを判断します。Amazon S3 は、リクエストを処理するために使用するバケットを決定した後、そのバケットにリクエストを送信し、マルチリージョンアクセスポイントのホスト名を使用してオペレーションを記録します。この例では、Amazon S3 がリクエストを my-bucket-aps1 にリレーした場合、ログはホスト名 mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com を使用して、my-bucket-aps1 から、my-image.jpg に対して成功した GET リクエストを反映します。

> **Important**
>
> マルチリージョンアクセスポイントは、基になるバケットのデータコンテンツを認識しません。そのため、リクエストを受け取ったバケットには、リクエストされたデータが含まれていない可能性があります。例えば、Amazon S3 が my-bucket-euc1 バケットが最も低いと判断した場合、ログはホスト名 mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com を使用して、my-bucket-euc1 から my-image.jpg に対して失敗した GET リクエストを反映します。リクエストが代わりに my-bucket-usw2 にルーティングされると、ログは成功した GET リクエストを表示します。

Amazon S3 サーバーのアクセスログの詳細については、「サーバーアクセスログによるリクエストのログ記録」を参照してください。AWS CloudTrail の詳細については、AWS CloudTrail ユーザーガイド の AWS CloudTrail とは を参照してください。

マルチリージョンアクセスポイント管理 API オペレーションに対するリクエストのモニタリングとログ記録

Amazon S3 は、CreateMultiRegionAccessPoint および GetMultiRegionAccessPointPolicy など、マルチリージョンアクセスポイントを管理するためのいくつかの API オペレーションを提供します。AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して、これらの API オペレーションにリクエストを

実行すると、Amazon S3 はこれらのリクエストを非同期的に処理します。リクエストに対して適切な許可がある場合、Amazon S3 は、リクエストに対してトークンを返します。このトークンは、`DescribeAsyncOperation` が実行中の非同期オペレーションのステータスを表示するのに役立ちます。Amazon S3 は、`DescribeAsyncOperation` リクエストを同期的に処理します。非同期リクエストのステータスを表示するには、Amazon S3 コンソール、AWS CLI、SDK、または REST API を使用できます。

> **Note**
>
> コンソールには、過去 14 日以内に行われた非同期リクエストのステータスのみが表示されます。古いリクエストのステータスを表示するには、AWS CLI、SDK、または REST API を使用します。

非同期管理オペレーションは、次の状態のいずれかになります。

NEW

Amazon S3 がリクエストを受け取り、オペレーションの実行準備を進めています。

IN_PROGRESS

Amazon S3 は現在、オペレーションを実行しています。

SUCCESS

オペレーションは成功しました。応答には、`CreateMultiRegionAccessPoint` リクエストのマルチリージョンアクセスポイントのエイリアスなど、関連情報が含まれます。

FAILED

このオペレーションは失敗しました。応答には、リクエストが失敗した理由を示すエラーメッセージが含まれます。

AWS CloudTrail とマルチリージョンアクセスポイントの使用

AWS CloudTrail を使用して、AWS インフラストラクチャ全体のアカウントアクティビティを表示、検索、ダウンロード、アーカイブ、分析、応答できます。マルチリージョンのアクセスポイントと CloudTrail の口グ記録では、次のことを識別できます。

- 誰が、または何が、どのアクションを実行したか
- どのリソースに基づいて処理が行われたか

- イベントが発生した時刻
- イベントに関するその他の詳細

このログ記録情報は、マルチリージョンアクセスポイントを通じて発生したアクティビティの分析と対応に役立ちます。

マルチリージョンアクセスポイントの AWS CloudTrail のセットアップ方法

マルチリージョンアクセスポイントの作成または保守に関するオペレーションで CloudTrail ログを有効にするには、米国西部 (オレゴン) リージョン内のイベントを記録するように CloudTrail ログを設定する必要があります。リクエストの実行時にどのリージョンにいるか、またはマルチリージョンアクセスポイントがどのリージョンをサポートしているかに関係なく、この方法でログ記録設定をセットアップする必要があります。マルチリージョンアクセスポイントを作成または維持するためのすべてのリクエストは、米国西部 (オレゴン) リージョンを通じてルーティングされます。このリージョンを既存の追跡に追加するか、このリージョンとマルチリージョンアクセスポイントに関連付けられたすべてのリージョンを含む新しい追跡を作成することをお勧めします。

Amazon S3 は、マルチリージョンアクセスポイントを介して行われたすべてのリクエストと、アクセスポイントを管理する API に対するリクエスト (`CreateMultiRegionAccessPoint` や `GetMultiRegionAccessPointPolicy` など) をログに記録します。マルチリージョンアクセスポイントを介してこれらのリクエストを記録すると、マルチリージョンアクセスポイントのホスト名を使用して AWS CloudTrail ログに表示されます。例えば、マルチリージョンのアクセスポイントを介してバケットにエイリアス `mfzwi23gnjvgw.mrap` でリクエストを行った場合、CloudTrail ログのエントリのホスト名は `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com` になります。

マルチリージョンアクセスポイントは、リクエストを最も近いバケットにルーティングします。この動作のため、マルチリージョンアクセスポイントの CloudTrail ログを見ると、基となるバケットに対してリクエストが行われていることがわかります。ここうしたリクエストの一部は、マルチリージョンアクセスポイントを経由しない、バケットへの直接リクエストである場合があります。トラフィックを確認するときは、この点に注意してください。バケットがマルチリージョンアクセスポイントにある場合でも、マルチリージョンアクセスポイントを経由しなくても、そのバケットに対してリクエストを直接行うことができます。

マルチリージョンアクセスポイントの作成と管理には、非同期イベントがあります。非同期リクエストでは、CloudTrail ログに完了イベントはありません。リクエストの再試行の詳細については、「マルチリージョンアクセスポイント管理 API オペレーションに対するリクエストのモニタリングとログ記録」を参照してください。

AWS CloudTrail の詳細については、AWS CloudTrail ユーザーガイド の AWS CloudTrail とは を参照してください。

Amazon S3 のセキュリティ

AWS では、クラウドのセキュリティが最優先事項です。AWS のお客様は、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャから利点を得られます。

セキュリティは、AWS とお客様の間の共有責任です。責任共有モデルでは、この責任がクラウドのセキュリティおよびクラウド内のセキュリティとして説明されています。

クラウドのセキュリティ

AWS は AWS クラウド 内で AWS サービスを実行するインフラストラクチャを保護する責任を担います。また、AWS は、ユーザーが安全に使用できるサービスも提供します。セキュリティの有効性は、AWS コンプライアンスプログラムの一環として、サードパーティーの監査機関によって定期的にテストおよび検証されています。Amazon S3 に適用されるコンプライアンスプログラムについては、AWSコンプライアンスプログラムによるターゲット範囲の のサービスを参照してください。

クラウド内のセキュリティ

お客様の責任は、お客様が使用する AWS のサービスによって決まります。お客様は、データの機密性、組織の要件、および適用法令と規制などのその他要因に対する責任も担います。Amazon S3 において、お客様は以下の事項について責任を負います。

- オブジェクトの所有権 および暗号化 を含むデータの管理。
- アセットの分類。
- IAM ロール およびその他のサービス設定を使用し、適切な権限を適用したデータへのアクセスの管理 。
- AWS CloudTrail または Amazon S3 の Amazon GuardDuty 等のディテクティブコントロールの有効化。

このドキュメントは、Amazon S3 を使用する際に適用される責任共有モデルについての理解の助けとなることを目的としています。以下のトピックでは、セキュリティとコンプライアンスの目的を満たすように Amazon S3 を設定する方法について説明します。また、Amazon S3 のリソースのモニタリングや保護に役立つ AWS の他のサービスを使用する方法についても説明します。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

トピック

Amazon S3 におけるデータ保護

Amazon S3 は、ミッションクリティカルで重要なデータストレージのために設計された、高い耐久性を備えたストレージインフラストラクチャです。S3 標準、S3 Intelligent-Tiering、S3 標準 — IA、S3 Glacier インスタント検索、S3 Glacier フレキシブル検索、および S3 Glacier Deep Archive は、AWS リージョン の最低 3 つのアベイラビリティーゾーンにわたって複数のデバイスにオブジェクトを冗長して保存します。アベイラビリティーゾーンは、AWS リージョン の冗長電源、ネットワーク、および接続を備えた 1 つ以上の個別のデータセンターです。アベイラビリティーゾーンは、他のアベイラビリティーゾーンから意味のある距離 (キロメートル) で物理的に分離されていますが、すべてが互いに 100 km (60 マイル)以内にあります。S3 1 ゾーンストレージクラスは、単一

のアベイラビリティーゾーン内の複数のデバイスにデータを冗長に保存します。これらのサービスは、失われた冗長性を迅速に検出して修復することにより、同時デバイス障害を処理するように設計されており、また、チェックサムを使用してデータの整合性を定期的に検証します。

Amazon S3 Standard ストレージには以下の特徴があります。

- Amazon S3 サービスレベルアグリーメントに裏づけられています。
- 1 年間に 99.999999999% の堅牢性と、99.99% のオブジェクトの可用性を提供するよう設計されています。
- S3 Standard、S3 Intelligent-Tiering、S3 標準 — IA、S3 Glacier インスタント検索、S3 Glacier フレキシブル検索、S3 Glacier Deep Archive は、Amazon S3 アベイラビリティーゾーン全体が失われた場合にデータを維持するように設計されています。

Amazon S3 は、バージョニングを使用してデータ保護を強化しています。バージョニングを使用して、Amazon S3 バケットに格納されたあらゆるオブジェクトのあらゆるバージョンを、格納、取得、復元することができます。バージョニングを使用すれば、意図しないユーザーアクションからもアプリケーション障害からも、簡単に復旧できます。デフォルトでは、リクエストは最も新しく書き込まれたバージョンを取得します。リクエストでオブジェクトのバージョンを指定することにより、古いバージョンのオブジェクトを取得できます。

S3 バージョニングとは別に、Amazon S3 オブジェクトロックと S3 レプリケーションを使用してデータを保護することもできます。詳細については、「チュートリアル: S3 バージョニング、S3 オブジェクトロック、S3 レプリケーションを使用して Amazon S3 上のデータを誤って削除したり、アプリケーションのバグから保護したりする」を参照してください。

データ保護目的の場合、AWS アカウント 認証情報を保護して AWS Identity and Access Management で個々のユーザーアカウントをセットアップし、そのユーザーに各自の職務を果たすために必要な許可のみが付与されるようにすることをお勧めします。

コマンドラインインターフェイスまたは API を使用して AWS にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。使用可能な FIPS エンドポイントの詳細については、連邦情報処理規格 (FIPS) 140-2 を参照してください。

以下のセキュリティのベストプラクティスも Amazon S3 でのデータ保護に対処します。

- Implement server-side encryption
- Enforce encryption of data in transit
- Consider using Macie with Amazon S3

- Identify and audit all your Amazon S3 buckets
- Monitor Amazon Web Services security advisories

暗号化によるデータの保護

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

データ保護とは、転送時 (Amazon S3 との間でデータを送受信するとき)と保存時 (Amazon S3 データセンター内のディスクに格納されているとき) にデータを保護することを指します。Secure Socket Layer/Transport Layer Security (SSL/TLS) またはクライアント側の暗号化を使用して、転送中のデータを保護できます。Amazon S3 で保管時のデータを保護するには、次のようなオプションがあります。

- サーバー側の暗号化 – Amazon S3 は、AWS データセンターのディスクに保存する前にオブジェクトを暗号化し、ダウンロードするときにオブジェクトを復号化します。

  すべての Amazon S3 バケットにはデフォルトで暗号化が設定されており、S3 バケットにアップロードされたすべての新しいオブジェクトは保存時に自動的に暗号化されます。Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) は、Amazon S3 のすべてのバケットでのデフォルトの暗号化設定です。別のタイプの暗号化を使用するには、S3 PUT リクエストで使用するサーバー側の暗号化のタイプを指定するか、宛先バケットにデフォルトの暗号化設定を設定できます。

  PUT リクエストで別の暗号化タイプを指定する場合は、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS)、または顧客提供のキーによるサーバー側の暗号化 (SSE-C) を使用できます。

宛先バケットに別のデフォルト暗号化設定を設定する場合は、SSE-KMS または DSSE-KMS を使用できます。

サーバー側の暗号化の詳細については、「サーバー側の暗号化によるデータの保護」を参照してください。

サーバー側の暗号化を設定するには、以下を参照してください。

- Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) の指定
- AWS KMS (SSE-KMS) によるサーバー側の暗号化の指定
- the section called “DSSE-KMS の指定”
- お客様が用意したキーによるサーバー側の暗号化 (SSE-C) の指定

- クライアント側の暗号化 – クライアント側でデータを暗号化し、暗号化したデータを Amazon S3 にアップロードします。この場合、暗号化プロセス、暗号化キー、関連ツールはお客様が管理してください。

  クライアント側の暗号化を設定するには、クライアント側の暗号化を使用したデータの保護 を参照してください。

ストレージバイトの何パーセントが暗号化されているかを確認するには、Amazon S3 ストレージレンズメトリクスを使用できます。S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージ分析機能です。詳細については、「S3 Storage Lens を使用したストレージのアクティビティと使用状況の評価」を参照してください。メトリクスの完全なリストについては、「S3 ストレージレンズメトリクスに関する用語集」を参照してください。

サーバー側の暗号化とクライアント側の暗号化の詳細については、以下のトピックを参照してください。

トピック

- サーバー側の暗号化によるデータの保護
- クライアント側の暗号化を使用したデータの保護

サーバー側の暗号化によるデータの保護

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

サーバー側の暗号化とは、データを受信するアプリケーションまたはサービスによって、送信先でデータを暗号化することです。Amazon S3 は、AWS データセンターのディスクに書き込まれるときにデータをオブジェクトレベルで暗号化し、お客様がデータにアクセスするときに復号します。リクエストが認証され、お客様がアクセス許可を持っていれば、オブジェクトが暗号化されているかどうかに関係なく同じ方法でアクセスできます。例えば、署名付き URL を使用してオブジェクトを共有する場合、その署名付き URL は、オブジェクトが暗号化されているかどうかに関係なく同じように動作します。また、バケット内のオブジェクトを一覧表示すると、リスト API オペレーションは、オブジェクトが暗号化されているかどうかに関係なく、すべてのオブジェクトのリストを返します。

すべての Amazon S3 バケットにはデフォルトで暗号化が設定されており、S3 バケットにアップロードされたすべての新しいオブジェクトは保存時に自動的に暗号化されます。Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) は、Amazon S3 のすべてのバケットでのデフォルトの暗号化設定です。別のタイプの暗号化を使用するには、S3 PUT リクエストで使用するサーバー側の暗号化のタイプを指定するか、宛先バケットにデフォルトの暗号化設定を設定できます。

PUT リクエストで別の暗号化タイプを指定する場合は、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS)、または顧客提供のキーによるサーバー側の暗号化 (SSE-C) を使用できます。宛先バケットに別のデフォルト暗号化設定を設定する場合は、SSE-KMS または DSSE-KMS を使用できます。

> **Note**
>
> 異なる種類のサーバー側暗号化を同時に同じオブジェクトに適用することはできません。

既存のオブジェクトを暗号化する必要がある場合は、S3 バッチオペレーションと S3 インベントリを使用してください。詳細については、「Encrypting objects with Amazon S3 Batch Operations」(Amazon S3 バッチオペレーションよるオブジェクトの暗号化) および Amazon S3 オブジェクトでの大規模なバッチ操作の実行 を参照してください。

サーバー側の暗号化には、暗号化キーの管理方法の選択と適用する暗号化レイヤーの数に応じて、相互に排他的な 4 つのオプションがあります。

Amazon S3 マネージドキーを用いたサーバー側の暗号化 (SSE-S3)

すべての Amazon S3 バケットには、デフォルトで暗号化が設定されています。サーバー側の暗号化のデフォルトのオプションは、Amazon S3 マネージド キー (SSE-S3) を使用しています。一意のキーで各オブジェクトを暗号化します。追加の保護措置として、SSE-S3 は定期的にローテーションされるルートキーを使ってキーそのものを暗号化します。SSE-S3 は、利用可能な最強のブロック暗号の 1 つである 256 ビットの 高度暗号化規格 (AES-256) を使用して、データを暗号化します。詳細については、「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)」を参照してください。

AWS Key Management Service (AWS KMS) キー (SSE-KMS) によるサーバー側の暗号化

AWS KMS keys (SSE-KMS) によるサーバー側の暗号化は、AWS KMS サービスと Amazon S3 の統合によって提供されます。AWS KMS を使用すると、キーをより細かく制御できます。例えば、個別のキーを表示し、制御ポリシーを編集して、AWS CloudTrail のキーに従うことができます。さらに、カスタマーマネージドキーを作成および管理したり、ユーザー、サービス、およびリージョンに固有の AWS マネージドキー を使用したりできます。詳細については、「AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用」を参照してください。

AWS Key Management Service (AWS KMS) キーによる二層式サーバー側の暗号化 (DSSE-KMS)

AWS KMS keys による二層式サーバー側の暗号化 (DSSE-KMS) は、SSE-KMS と似ていますが、DSSE-KMS では 1 つのレイヤーの代わりに 2 つの個別のオブジェクトレベルの暗号化レイヤーを適用します。両方の暗号化レイヤーがサーバーサイドでオブジェクトに適用されるため、コンプライアンス要件を満たす暗号化方法を使用しながら、幅広い AWS のサービス とツールを使用して S3 内のデータを分析できます。詳細については、「AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS) の使用」を参照してください。

顧客提供のキーを用いたサーバー側の暗号化 (SSE-C)。

顧客提供キーによるサーバー側の暗号化 (SSE-C) では、お客様が暗号化キーを管理し、Amazon S3 はディスクに書き込む際の暗号化とオブジェクトにアクセスする際の復号を管理します。詳細については、「お客様が指定したキーによるサーバー側の暗号化 (SSE-C) の使用」を参照してください。

Amazon S3 では、すべての新しいオブジェクトが自動的に暗号化されるようになりました

Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。256 ビットの高度暗号化規格 (AES-256) を使用する SSE-S3 は、すべての新しいバケットと、デフォルトの暗号化がまだ設定されていない既存の S3 バケットに自動的に適用されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface (AWS CLI) と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。

以下のセクションでは、この更新に関する質問に答えます。

Amazon S3 は、既にデフォルトの暗号化が設定されている既存のバケットのデフォルトの暗号化設定を変更しますか?

いいえ。SSE-S3 または AWS Key Management Service (AWS KMS) キーを使用したサーバー側の暗号化 (SSE-KMS) が設定されている既存のバケットのデフォルトの暗号化設定は変更されません。バケットのデフォルトの暗号化動作を設定する方法の詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。SSE-S3 および SSE-KMS 暗号化設定に関する詳細は、「サーバー側の暗号化によるデータの保護」を参照してください。

デフォルトの暗号化が設定されていない既存のバケットでもデフォルトの暗号化は有効になりますか?

はい。Amazon S3 は、暗号化されていない既存のバケットすべてにデフォルトの暗号化を設定して、これらのバケットにアップロードされる新しいオブジェクトの暗号化の基本レベルとして、S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) を適用するようになりました。既存の暗号化されていないバケットに既に存在するオブジェクトは、自動的に暗号化されません。

新しいオブジェクトアップロードのデフォルトの暗号化ステータスを表示する方法を教えてください。

現在、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface (AWS CLI) と AWS SDK の追加の Amazon S3 API レスポンスヘッダーで、新しいオブジェクトのアップロードのデフォルトの暗号化ステータスを確認できます。

- CloudTrail イベントを確認するには、「AWS CloudTrail ユーザーガイド」の「CloudTrail コンソールで CloudTrail イベントを表示する」を参照してください。CloudTrail ログは、Amazon S3 への PUT および POST リクエストの API トラッキングを提供します。バケット内のオブジェクトを暗号化するためにデフォルトの暗号化が使用されている場合、PUT および POST API リクエストの CloudTrail ログには、名前と値のペアとして次のフィールドが含まれます。`"SSEApplied":"Default_SSE_S3"`
- S3 インベントリにおける新しいオブジェクトのアップロードの自動暗号化ステータスを確認するには、[Encryption] (暗号化) メタデータフィールドを含むように S3 インベントリレポートを設定し、レポート内の新しいオブジェクトごとに暗号化ステータスを確認します。詳細については、「Amazon S3 インベントリの設定」を参照してください。
- S3 ストレージレンズにおける新しいオブジェクトのアップロードの自動暗号化ステータスを確認するには、S3 ストレージレンズダッシュボードを設定し、ダッシュボードの [Data protection] (データ保護) カテゴリで、[Encrypted bytes] (暗号化されたバイト数) メトリクスと [Encrypted object count] (暗号化されたオブジェクト数) メトリクスを確認します。詳細については、Amazon S3 Storage Lens ダッシュボードの作成およびダッシュボードで S3 Storage Lens のメトリクスを表示するを参照してください。
- Amazon S3 コンソールでバケットレベルの自動暗号化ステータスを確認するには、Amazon S3 コンソールで Amazon S3 バケットのデフォルトの暗号化を確認します。詳細については、「デフォルトの暗号化の設定」を参照してください。
- 自動暗号化ステータスを AWS Command Line Interface (AWS CLI) および AWS SDK の追加の Amazon S3 API レスポンスヘッダー `x-amz-server-side-encryption` として表示するには、PutObject や GetObject などのオブジェクトアクション API を使用するときにレスポンスヘッダーを確認してください。

この変更を利用するには何をすればよいですか?

既存のアプリケーションに変更を加える必要はありません。すべてのバケットでデフォルトの暗号化が有効になっているため、Amazon S3 にアップロードされたすべての新しいオブジェクトは自動的に暗号化されます。

バケットに書き込まれる新しいオブジェクトの暗号化を無効にすることはできますか?

いいえ。SSE-S3 は、バケットにアップロードされるすべての新しいオブジェクトに適用される新しい基本レベルの暗号化です。新しいオブジェクトアップロードの暗号化を無効にすることはできなくなりました。

私の請求は影響を受けますか?

いいえ。SSE-S3 を使用したデフォルトの暗号化は、追加料金なしで利用できます。通常どおり、ストレージ、リクエスト、その他の S3 機能の料金が請求されます。料金については、「Amazon S3 の料金」を参照してください。

Amazon S3 は、暗号化されていない既存のオブジェクトを暗号化しますか?

いいえ。 2023 年 1 月 5 日以降、Amazon S3 は新しいオブジェクトのアップロードのみを自動的に暗号化します。既存のオブジェクトを暗号化するには、S3 Batch オペレーションを使用してオブジェクトの暗号化されたコピーを作成します。これらの暗号化されたコピーには、既存のオブジェクトデータと名前が保持され、指定した暗号化キーを使用して暗号化されます。詳細については、「AWS ストレージブログ」の「Amazon S3 バッチオペレーションによるオブジェクトの暗号化」を参照してください。

このリリース以前は、バケットの暗号化を有効にしていませんでした。オブジェクトへのアクセス方法を変更する必要がありますか。

いいえ。SSE-S3 を使用したデフォルトの暗号化は、Amazon S3 に書き込まれるときにデータを自動的に暗号化し、お客様がデータにアクセスするときに復号します。自動的に暗号化されたオブジェクトへのアクセス方法に変更はありません。

クライアント側の暗号化オブジェクトにアクセスする方法を変更する必要がありますか。

いいえ。Amazon S3 にアップロードされる前に暗号化されたクライアント側の暗号化されたオブジェクトはすべて、暗号化された暗号文オブジェクトとして Amazon S3 に送られます。これらのオブジェクトには SSE-S3 暗号化レイヤーが追加されます。クライアント側の暗号化オブジェクトを使用するワークロードでは、クライアントサービスや認証設定を変更する必要はありません。

> **Note**
>
> 更新されたバージョンの AWS プロバイダーを使用していない HashiCorp Terraform ユーザーは、お客様定義の暗号化設定なしで新しい S3 バケットを作成した後、予期しないドリフトが発生する可能性があります。このドリフトを避けるには、お使いの Terraform AWS プ

ロバーダーバージョンを次のいずれかのバージョンに更新してください。4.x のいずれかのリリース、3.76.1、2.70.4。

Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

Amazon S3 バケットへの新しいオブジェクトのアップロードはすべて、デフォルトでは Amazon S3 マネージドキーによる暗号化 (SSE-S3) で暗号化されます。

サーバー側の暗号化は、保管中のデータを保護します。Amazon S3 は、一意のキーで各オブジェクトを暗号化します。追加の保護措置として、S3 は定期的にローテーションされるキーを使ってキーそのものを暗号化します。Amazon S3 サーバー側の暗号化では、256 ビットの高度暗号化標準 Galois/Counter Mode (AES-GCM) を使用して、アップロードされたすべてのオブジェクトを暗号化します。

Amazon S3 が管理するキーによるサーバー側の暗号化 (SSE-S3) を使用するために、追加でかかる料金はありません。ただし、デフォルトの暗号化機能を設定するためのリクエストには、標準 Amazon S3 リクエスト料金がかかります。料金については、Amazon S3 の料金を参照してください。

Amazon S3 マネージドキーのみを使用してデータアップロードを暗号化する必要がある場合は、次のバケットポリシーを使用できます。例えば、次のバケットポリシーは、サーバー側の暗号化を要求する `x-amz-server-side-encryption` ヘッダーがリクエストに含まれない限り、オブジェクトをアップロードする許可を拒否します。

```
{
```

```
  "Version": "2012-10-17",
  "Id": "PutObjectPolicy",
  "Statement": [
    {
      "Sid": "DenyObjectsThatAreNotSSES3",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::example-s3-bucket/*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "AES256"
        }
      }
    }
   ]
}
```

> **Note**
>
> サーバー側の暗号化では、オブジェクトのメタデータではなく、オブジェクトデータのみが暗号化されます。

サーバー側の暗号化での API サポート

すべての Amazon S3 バケットにはデフォルトで暗号化が設定されており、S3 バケットにアップロードされたすべての新しいオブジェクトは保存時に自動的に暗号化されます。Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) は、Amazon S3 のすべてのバケットでのデフォルトの暗号化設定です。別のタイプの暗号化を使用するには、S3 PUT リクエストで使用するサーバー側の暗号化のタイプを指定するか、宛先バケットにデフォルトの暗号化設定を設定できます。

PUT リクエストで別の暗号化タイプを指定する場合は、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS)、または顧客提供のキーによるサーバー側の暗号化 (SSE-C) を使用できます。宛先バケットに別のデフォルト暗号化設定を設定する場合は、SSE-KMS または DSSE-KMS を使用できます。

オブジェクト作成の REST API を使用してサーバー側の暗号化を設定するには、`x-amz-server-side-encryption` リクエストヘッダーを提供します。REST API の詳細については、REST API の使用 を参照してください。

以下の Amazon S3 API でこのヘッダーがサポートされています。

- PUT オペレーション – PUT API を使用してデータをアップロードする場合は、リクエストヘッダーを指定します。詳細については、「PutObject」を参照してください。
- マルチパートアップロードの開始 — マルチパートアップロード API オペレーションを使用して大きいオブジェクトをアップロードするときに、開始リクエストでヘッダーを指定します。詳細については、Initiate Multipart Upload を参照してください。
- COPY オペレーション – オブジェクトをコピーする場合、ソースオブジェクトとターゲットオブジェクトがあります。詳細については、PUT Object - Copy を参照してください。

> **Note**
>
> POST オペレーションを使用してオブジェクトをアップロードする場合は、リクエストヘッダーを指定する代わりに、フォームフィールドで同じ情報を指定します。詳細については、POST Object を参照してください。

AWS SDK にも、サーバー側の暗号化を要求するために使用できるラッパー API があります。また、AWS Management Console を使用して、オブジェクトをアップロードしてサーバー側の暗号化を要求することもできます。

詳細については、AWS Key Management Service デベロッパーガイドの「AWS KMS の概念」を参照してください。

トピック

- Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) の指定

Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) の指定

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレン

ズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

すべての Amazon S3 バケットにはデフォルトで暗号化が設定されており、S3 バケットにアップロードされたすべての新しいオブジェクトは保存時に自動的に暗号化されます。Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) は、Amazon S3 のすべてのバケットでのデフォルトの暗号化設定です。別のタイプの暗号化を使用するには、S3 PUT リクエストで使用するサーバー側の暗号化のタイプを指定するか、宛先バケットにデフォルトの暗号化設定を設定できます。

PUT リクエストで別の暗号化タイプを指定する場合は、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS)、または顧客提供のキーによるサーバー側の暗号化 (SSE-C) を使用できます。宛先バケットに別のデフォルト暗号化設定を設定する場合は、SSE-KMS または DSSE-KMS を使用できます。

S3 コンソール、REST API、AWS SDK、および AWS Command Line Interface (AWS CLI) を使用して SSE-S3 を指定できます。詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。

S3 コンソールの使用

このトピックでは、AWS Management Console を使用してオブジェクトの暗号化のタイプを設定または変更する方法について説明します。コンソールを使用してオブジェクトをコピーする場合、Amazon S3 はオブジェクトを現状のままコピーします。つまり、ソースが暗号化されている場合、ターゲットオブジェクトも暗号化されます。コンソールを使用して、オブジェクトの暗号化を追加または変更することもできます。

Note

- オブジェクトの暗号化を変更すると、新しいオブジェクトが作成され、古いオブジェクトが置き換えられます。S3 バージョニングが有効になっている場合は、オブジェクトの新しいバージョンが作成され、既存のオブジェクトが古いバージョンになります。また、プロパティを変更するロールは、新しいオブジェクト (またはオブジェクトのバージョン) の所有者になります。
- ユーザー定義タグを持つオブジェクトの暗号化タイプを変更する場合は、`s3:GetObjectTagging` アクセス許可が必要です。ユーザー定義タグが

なく、サイズが 16 MB を超えるオブジェクトの暗号化タイプを変更する場合は、`s3:GetObjectTagging` アクセス許可も必要です。

ターゲットバケットポリシーによって `s3:GetObjectTagging` アクションが拒否されると、オブジェクトの暗号化タイプは更新されますが、ユーザー定義タグはオブジェクトから削除され、エラーが発生します。

オブジェクトの暗号化を変更するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [Buckets] (バケット) リストで、オブジェクトが含まれるバケットの名前を選択します。
4. [オブジェクト] リストで、暗号化を追加または変更するオブジェクトの名前を選択します。

   オブジェクトの詳細ページが開き、オブジェクトのプロパティを表示するいくつかのセクションが表示されます。
5. [プロパティ] タブを選択します。
6. 「サーバー側の暗号化設定」セクションまでスクロールし、[編集] を選択します。
7. [暗号化設定] で、[バケットのデフォルトの暗号化設定を使用する] または [バケットのデフォルトの暗号化設定を上書きする] を選択します。
8. [デフォルトの暗号化にバケット設定を上書きする] を選択した場合は、次の暗号化設定を設定します。

   - [暗号化タイプ] で、[Amazon S3 マネージドキー (SSE-S3)] を選択します。SSE-S3 は、最強のブロック暗号の 1 つである 256 ビットの 高度暗号化規格 (AES-256) を使用して、各オブジェクトを暗号化します。詳細については、「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)」を参照してください。
9. [Save changes] (変更の保存) をクリックします。

> **Note**
>
> このアクションは、指定されたすべてのオブジェクトに暗号化を適用します。フォルダを暗号化する場合は、保存オペレーションが完了するのを待ってから、フォルダに新しいオブジェクトを追加します。

REST API の使用

オブジェクト作成時、つまり、新しいオブジェクトをアップロードするときや既存オブジェクトのコピーを作成するときに、リクエストに `x-amz-server-side-encryption` ヘッダーを追加することで、Amazon S3 マネージドキー (SSE-S3) を使用してデータを暗号化するかどうかを指定することができます。ヘッダーの値を、Amazon S3 がサポートする暗号化アルゴリズム `AES256` に設定します。Amazon S3 によりレスポンスヘッダー `x-amz-server-side-encryption` が返されるため、SSE-S3 を使用してオブジェクトが保存されたことを確認できます。

次の REST アップロード API オペレーションは、`x-amz-server-side-encryption` リクエストヘッダーを受け付けます。

- PUT Object
- PUT Object – Copy
- POST Object
- Initiate Multipart Upload

マルチパートアップロード API オペレーションを使用して大容量のオブジェクトをアップロードするときに、`x-amz-server-side-encryption` ヘッダーを Initiate Multipart Upload リクエストに追加することでサーバー側の暗号化を指定することができます。既存のオブジェクトをコピーする際は、コピー元オブジェクトが暗号化されているかどうかに関係なく、明示的にサーバー側の暗号化を要求しない限り、コピー先オブジェクトは暗号化されません。

次の REST API オペレーションのレスポンスヘッダーは、オブジェクトが SSE-S3 を使用して保存されているときに `x-amz-server-side-encryption` ヘッダーを返します。

- PUT Object
- PUT Object – Copy
- POST Object
- Initiate Multipart Upload

- Upload Part
- Upload Part – Copy
- Complete Multipart Upload
- Get Object
- Head Object

> **Note**
>
> オブジェクトで SSE-S3 を使用している場合、または、HTTP ステータスコード 400 (BadRequest) エラーが発生する場合、暗号化リクエストヘッダーを `GET` リクエストおよび `HEAD` リクエストに対して送信しないでください。

AWS SDK の使用

AWS SDK を使用して、Amazon S3 マネージド暗号化キー (SSE-S3) を使用するように Amazon S3 にリクエストできます。このセクションでは、AWS SDK を複数の言語で使用する場合の例を示します。他の SDK については、「サンプルコードとライブラリ」を参照してください。

Java

AWS SDK for Java を使用してオブジェクトをアップロードしたら、SSE-S3 を使用してデータを暗号化します。サーバー側の暗号化を要求するには、`ObjectMetadata` の `PutObjectRequest` プロパティを使用して `x-amz-server-side-encryption` リクエストヘッダーを設定します。`AmazonS3Client` の `putObject()` メソッドを呼び出すと、Amazon S3 はデータを暗号化して保存します。

さらに、マルチパートアップロード API オペレーションを使用して、オブジェクトのアップロード時に、SSE-S3 暗号化をリクエストすることもできます。

- 高レベルマルチパートアップロード API オペレーションを使用するときに、`TransferManager` メソッドを使用して、アップロード時にサーバー側の暗号化をオブジェクトに適用できます。パラメータとして `ObjectMetadata` を受け取る、どのアップロードメソッドを使用してもかまいません。詳細については、「マルチパートアップロードを使用したオブジェクトのアップロード」を参照してください。
- 低レベルマルチパートアップロード API オペレーションを使用する場合は、マルチパートアップロードの開始時にサーバー側の暗号化を指定します。`ObjectMetadata` メソッドを呼び出

して InitiateMultipartUploadRequest.setObjectMetadata() プロパティを追加します。詳細については、「AWS SDK の使用 (低レベル API)」を参照してください。

オブジェクトの暗号化状態を直接変更することはできません (暗号化されていないオブジェクトの暗号化、または暗号化されたオブジェクトの復号)。オブジェクトの暗号化状態を変更するには、オブジェクトのコピーを作成し、コピーの目的の暗号化状態を指定してから、元のオブジェクトを削除します。Amazon S3 は、明示的にサーバー側の暗号化を要求した場合にのみ、コピーされたオブジェクトを暗号化します。Java API を通じて、コピーされたオブジェクトの暗号化を要求するには、ObjectMetadata プロパティを使用して、CopyObjectRequest でサーバー側の暗号化を指定します。

Example 例

次の例は、AWS SDK for Java を使用してサーバー側の暗号化を設定する方法を示しています。ここでは、以下のタスクの実行方法について説明します。

- SSE-S3 を使用して新しいオブジェクトをアップロードします。
- オブジェクトのコピーを作成して、オブジェクトの暗号化状態を変更する (この例では、以前に暗号化されていないオブジェクトを暗号化する)
- オブジェクトの暗号化状態を確認する

サーバーサイドの暗号化の詳細については、「REST API の使用」を参照してください。作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.internal.SSEResultBase;
import com.amazonaws.services.s3.model.*;

import java.io.ByteArrayInputStream;

public class SpecifyServerSideEncryption {
```

```
   public static void main(String[] args) {
       Regions clientRegion = Regions.DEFAULT_REGION;
       String bucketName = "*** Bucket name ***";
       String keyNameToEncrypt = "*** Key name for an object to upload and encrypt
***";
       String keyNameToCopyAndEncrypt = "*** Key name for an unencrypted object to
be encrypted by copying ***";
       String copiedObjectKeyName = "*** Key name for the encrypted copy of the
unencrypted object ***";

       try {
           AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                   .withRegion(clientRegion)
                   .withCredentials(new ProfileCredentialsProvider())
                   .build();

           // Upload an object and encrypt it with SSE.
           uploadObjectWithSSEEncryption(s3Client, bucketName, keyNameToEncrypt);

           // Upload a new unencrypted object, then change its encryption state
           // to encrypted by making a copy.
           changeSSEEncryptionStatusByCopying(s3Client,
                   bucketName,
                   keyNameToCopyAndEncrypt,
                   copiedObjectKeyName);
       } catch (AmazonServiceException e) {
           // The call was transmitted successfully, but Amazon S3 couldn't process
           // it, so it returned an error response.
           e.printStackTrace();
       } catch (SdkClientException e) {
           // Amazon S3 couldn't be contacted for a response, or the client
           // couldn't parse the response from Amazon S3.
           e.printStackTrace();
       }
   }

   private static void uploadObjectWithSSEEncryption(AmazonS3 s3Client, String
bucketName, String keyName) {
       String objectContent = "Test object encrypted with SSE";
       byte[] objectBytes = objectContent.getBytes();

       // Specify server-side encryption.
       ObjectMetadata objectMetadata = new ObjectMetadata();
       objectMetadata.setContentLength(objectBytes.length);
```

```
objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
       PutObjectRequest putRequest = new PutObjectRequest(bucketName,
               keyName,
               new ByteArrayInputStream(objectBytes),
               objectMetadata);

       // Upload the object and check its encryption status.
       PutObjectResult putResult = s3Client.putObject(putRequest);
       System.out.println("Object \"" + keyName + "\" uploaded with SSE.");
       printEncryptionStatus(putResult);
   }

   private static void changeSSEEncryptionStatusByCopying(AmazonS3 s3Client,
           String bucketName,
           String sourceKey,
           String destKey) {
       // Upload a new, unencrypted object.
       PutObjectResult putResult = s3Client.putObject(bucketName, sourceKey,
"Object example to encrypt by copying");
       System.out.println("Unencrypted object \"" + sourceKey + "\" uploaded.");
       printEncryptionStatus(putResult);

       // Make a copy of the object and use server-side encryption when storing the
       // copy.
       CopyObjectRequest request = new CopyObjectRequest(bucketName,
               sourceKey,
               bucketName,
               destKey);
       ObjectMetadata objectMetadata = new ObjectMetadata();

objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
       request.setNewObjectMetadata(objectMetadata);

       // Perform the copy operation and display the copy's encryption status.
       CopyObjectResult response = s3Client.copyObject(request);
       System.out.println("Object \"" + destKey + "\" uploaded with SSE.");
       printEncryptionStatus(response);

       // Delete the original, unencrypted object, leaving only the encrypted copy
in
       // Amazon S3.
       s3Client.deleteObject(bucketName, sourceKey);
       System.out.println("Unencrypted object \"" + sourceKey + "\" deleted.");
```

```
    }

    private static void printEncryptionStatus(SSEResultBase response) {
        String encryptionStatus = response.getSSEAlgorithm();
        if (encryptionStatus == null) {
            encryptionStatus = "Not encrypted with SSE";
        }
        System.out.println("Object encryption status is: " + encryptionStatus);
    }
}
```

.NET

オブジェクトをアップロードするときに、それを暗号化するように Amazon S3 に指定できます。既存のオブジェクトの暗号化状態を変更するには、オブジェクトのコピーを作成し、コピー元オブジェクトを削除します。デフォルトでは、ターゲットオブジェクトのサーバー側の暗号化を明示的に要求した場合、コピーオペレーションではターゲットのみが暗号化されます。`CopyObjectRequest` で SSE-S3 を指定するには、以下を追加します。

```
ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
```

オブジェクトのコピー方法を示す作業サンプルについては、AWS SDK の使用 を参照してください。

次の例では、オブジェクトをアップロードします。このリクエストでは、オブジェクトを暗号化するように Amazon S3 に指定します。次に、オブジェクトのメタデータを取得し、使用された暗号化方法を確認します。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SpecifyServerSideEncryptionTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for object created ***";
```

```
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            WritingAnObjectAsync().Wait();
        }

        static async Task WritingAnObjectAsync()
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
                };

                var putResponse = await client.PutObjectAsync(putRequest);

                // Determine the encryption state of an object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };
                GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;

                Console.WriteLine("Encryption method used: {0}",
objectEncryption.ToString());
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
```

```
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

このトピックでは、バージョン 3 の AWS SDK for PHP のクラスを使用して、Amazon S3 にアップロードするオブジェクトに SSE-S3 を追加する方法を示します。AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

オブジェクトを Amazon S3 にアップロードするには、Aws\S3\S3Client::putObject() メソッドを使用します。アップロードリクエストに x-amz-server-side-encryption リクエストヘッダーを追加するには、ServerSideEncryption パラメータを指定して AES256 値を設定します。以下に例を示します。サーバー側の暗号化のリクエストについては、REST API の使用 を参照してください。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

// $filepath should be an absolute path to a file on disk.
$filepath = '*** Your File Path ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Upload a file with server-side encryption.
$result = $s3->putObject([
    'Bucket'               => $bucket,
    'Key'                  => $keyname,
    'SourceFile'           => $filepath,
    'ServerSideEncryption' => 'AES256',
```

```
]);
```

レスポンスとして、Amazon S3 は、x-amz-server-side-encryption ヘッダーを返します。このヘッダーには、オブジェクトのデータの暗号化に使用された暗号化アルゴリズムの値が設定されています。

マルチパートアップロード API オペレーションを使用して大容量のオブジェクトをアップロードする場合、以下のように、アップロードするオブジェクトに対して SSE-S3 を指定できます。

- 低レベルのマルチパートアップロード API オペレーションを使用する場合は、Aws\S3\S3Client::createMultipartUpload() メソッドを呼び出すときにサーバー側の暗号化を指定します。リクエストに x-amz-server-side-encryption ヘッダーを追加する場合は、値 array を使用して ServerSideEncryption パラメータの AES256 キーを指定します。低レベルのマルチパートアップロード API オペレーションの詳細については、「AWS SDK の使用 (低レベル API)」を参照してください。
- 高レベルのマルチパートアップロード API オペレーションを使用する場合は、CreateMultipartUpload API オペレーションの ServerSideEncryption パラメータを使用してサーバー側の暗号化を指定します。高レベルのマルチパートアップロード API オペレーションで setOption() メソッドを使用する例については、「マルチパートアップロードを使用したオブジェクトのアップロード」を参照してください。

以下の PHP コード例に示すように、Aws\S3\S3Client::headObject() メソッドを呼び出し、オブジェクトメタデータを取得して、既存のオブジェクトの暗号化状態を確認できます。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Check which server-side encryption algorithm is used.
$result = $s3->headObject([
    'Bucket' => $bucket,
```

```
    'Key'    => $keyname,
]);
echo $result['ServerSideEncryption'];
```

Aws\S3\S3Client::copyObject() メソッドを使用してオブジェクトのコピーを作成し、コピー元のオブジェクトを削除することで、既存のオブジェクトの暗号化状態を変更できます。デフォルトでは、copyObject() によってコピー先は暗号化されません。ただし、ServerSideEncryption パラメータで AES256 値を使用して、コピー先オブジェクトに対するサーバー側の暗号化を明示的にリクエストした場合は除きます。以下の PHP コード例では、オブジェクトのコピーを作成し、コピー先のオブジェクトにサーバー側の暗号化を追加しています。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
$sourceKeyname = '*** Your Source Object Key ***';

$targetBucket = '*** Your Target Bucket Name ***';
$targetKeyname = '*** Your Target Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Copy an object and add server-side encryption.
$s3->copyObject([
    'Bucket'               => $targetBucket,
    'Key'                  => $targetKeyname,
    'CopySource'           => "$sourceBucket/$sourceKeyname",
    'ServerSideEncryption' => 'AES256',
]);
```

詳細については、次のトピックを参照してください。

- Amazon S3 向け AWS SDK for PHP Aws\S3\S3Client クラスの場合
- AWS SDK for PHP ドキュメント

Ruby

AWS SDK for Ruby を使用してオブジェクトをアップロードするときは、SSE-S3 により保管時にオブジェクトを暗号化して保存することを指定できます。オブジェクトは、読み戻すときに自動的に復号されます。

次の AWS SDK for Ruby バージョン 3 のサンプルは、Amazon S3 にアップロードするファイルを保管時に暗号化するよう指定する方法を示しています。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end
```

```
run_demo if $PROGRAM_NAME == __FILE__
```

次のコード例は、既存のオブジェクトの暗号化状態を判定する方法を示しています。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? "no" :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Amazon S3 に保存されたオブジェクトにサーバー側の暗号化が使用されていない場合、メソッドは null を返します。

既存のオブジェクトの暗号化状態を変更するには、オブジェクトのコピーを作成し、コピー元オブジェクトを削除します。デフォルトでは、明示的にサーバー側の暗号化を要求しない限り、コピーメソッドはターゲットを暗号化しません。ターゲットオブジェクトの暗号化を要求するには、次の Ruby コードの例に示すように、オプションのハッシュ引数に server_side_encryption 値を指定します。このコード例は、オブジェクトをコピーし、コピーを SSE-S3 で暗号化する方法を示しています。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the target
 key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
```

```
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

AWS CLI の使用

AWS CLI を使用してオブジェクトをアップロードするときに SSE-S3 を指定するには、次の例を使用します。

```
aws s3api put-object --bucket example-s3-bucket1 --key object-key-name --server-side-
encryption AES256  --body file path
```

詳細については、AWS CLI リファレンスの put-object を参照してください。AWS CLI を使用してオブジェクトをコピーするときに SSE-S3 を指定するには、copy-object を参照してください。

AWS CloudFormation の使用

AWS CloudFormation を使用して暗号化を設定する例については、AWS CloudFormation ユーザーガイドの AWS::S3::Bucket ServerSideEncryptionRule トピックの「デフォルト暗号化を備えたバケットの作成」および「AWS KMS サーバー側の暗号化と S3 バケットキーを使用したバケットの作成」の例を参照してください。

AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

サーバー側の暗号化とは、データを受信するアプリケーションまたはサービスによって、送信先でデータを暗号化することです。

Amazon S3 は、新しいオブジェクトをアップロードするときに、Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) を自動的に有効にします。

特に指定しない限り、バケットはデフォルトで SSE-S3 を使用してオブジェクトを暗号化します。ただし、代わりにAWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS) を使用するようにバケットを設定することもできます。詳細については、「AWS KMS (SSE-KMS) によるサーバー側の暗号化の指定」を参照してください。

AWS KMS は、安全で可用性の高いハードウェアおよびソフトウェアを結合するサービスであり、クラウド向けに拡張されたキー管理システムを提供します。Amazon S3 は、AWS KMS (SSE-KMS) によるサーバー側の暗号化を使用して、S3 オブジェクトデータを暗号化します。また、オブジェクトに対して SSE-KMS をリクエストすると、オブジェクトのメタデータの一部としての S3 チェックサムが暗号化された形式で保存されます。チェックサムの詳細については、「オブジェクトの整合性をチェックする」を参照してください。

KMS キーを使用すると、AWS Management Console または AWS KMS API から AWS KMS を使用して、以下の操作を実行できます。

- KMS キーの作成、表示、編集、監視、有効化または無効化、更新、削除のスケジュールを一元的に行う。

- KMS キーの使用方法および使用者を制御するポリシーを定義する。
- 使用状況を監査して、正しく使用されていることを証明する。監査は AWS KMS API ではサポートされていますが、AWS KMSAWS Management Console ではサポートされていません。

AWS KMS のセキュリティ制御は、暗号化関連のコンプライアンス要件を満たすのに役立ちます。これらの KMS キーを使用して、Amazon S3 バケットのデータを保護できます。S3 バケットで SSE-KMS 暗号化を使用する場合、AWS KMS keys はバケットと同じリージョンに存在する必要があります。

AWS KMS keysを使用するための追加料金はかかります。詳細については、AWS Key Management Service デベロッパーガイドの AWS KMS key の概念およびAWS KMS の料金を参照してください。

アクセス許可

AWS KMS key を使用して暗号化されたオブジェクトを Amazon S3 にアップロードするには、キーに対する `kms:GenerateDataKey` の許可が必要です。AWS KMS keyを使用して暗号化されたオブジェクトをダウンロードするには、`kms:Decrypt` の許可が必要です。マルチパートアップロードに必要な AWS KMS アクセス許可については、[マルチパートアップロード API とアクセス許可]を参照してください。

> **Important**
>
> KMS キーポリシーで付与されているアクセス権限を慎重に確認します。カスタマーマネージド AWS KMS キーのポリシーのアクセス許可は常に、関連するキーアクションにアクセスする必要がある IAM プリンシパルと AWS サービスのみに限定する必要があります。詳細については、「AWS KMS のキーポリシー」を参照してください。

トピック

- AWS KMS keys
- Amazon S3 バケットキー
- サーバー側の暗号化の要求
- 暗号化コンテキスト
- AWS KMS 暗号化されたオブジェクトへのリクエストの送信
- AWS KMS (SSE-KMS) によるサーバー側の暗号化の指定

- Amazon S3 バケットキーを使用した SSE-KMS のコストの削減

AWS KMS keys

AWS KMS によるサーバー側暗号化 (SSE-KMS) を使用すると、デフォルトの AWS マネージドキーを使用するか、または、作成済みのカスタマーマネージドキーを指定できます。AWS KMS はエンベロープ暗号化をサポートします。S3 は、エンベロープ暗号化の AWS KMS 機能を使用してデータを保護します。エンベロープ暗号化は、データキーでプレーンテキストデータを暗号化してから、そのデータキーを KMS キーで暗号化する手法です。エンベロープ暗号化の仕組みの詳細については、AWS Key Management Service デベロッパーガイドの「エンベロープ暗号化」を参照してください。

カスタマーマネージドキーを指定しない場合、SSE-KMS で暗号化されたオブジェクトをバケットに初めて追加するとき、Amazon S3 によって自動的に AWS アカウント に AWS マネージドキー が作成されます。デフォルトでは、Amazon S3 はこの KMS キーを SSE-KMS に使用します。

> **Note**
>
> SSE-KMS を AWS マネージドキー で使用して暗号化されたオブジェクトは、クロスアカウントで共有できません。SSE-KMS データをアカウント間で共有する必要がある場合は、AWS KMS のカスタマーマネージドキーを使用する必要があります。

SSE-KMS にカスタマーマネージドキーを使用する場合は、SSE-KMS を設定する前に対称暗号化カスタマーマネージドキーを作成します。次に、バケット用に SSE-KMS を設定するときに、既存のカスタマーマネージドキーを指定します。対象暗号化キーの詳細については、「AWS Key Management Service デベロッパーガイド」の「対称暗号化 KMS キー」を参照してください。

カスタマーマネージドキーを独自に作成することで、より柔軟に制御を行えます。例えば、カスタマーマネージドキーを作成、ローテーション、無効化することができます。また、アクセスコントロールを定義し、データを保護するために使用するカスタマーマネージドキーを監査することもできます。カスタマーマネージドキーおよび AWS マネージドキーの詳細については、「AWS Key Management Service デベロッパーガイド」の「カスタマーキーおよび AWS キー」を参照してください。

> **Note**
>
> 標準の KMS キーとは異なり、外部のキーストアに保存されているカスタマーマネージドキーによるサーバー側の暗号化を使用する場合は、キーマテリアルの可用性と耐久性を確保

する責任があります。外部キーストアと、それらによって責任分担モデルがどのように変化するかについての詳細は、「AWS Key Management Service デベロッパーガイド」の「外部キーストア」を参照してください。

クロスアカウント操作での SSE-KMS 暗号化の使用

クロスアカウント操作で暗号化を使用する場合は、次の点に注意してください。

- AWS KMS key Amazon リソースネーム (ARN) またはエイリアスがリクエスト時またはバケットのデフォルト暗号化設定によって提供されない場合は、AWS マネージドキー (aws/s3) が使用されます。
- KMS キーと同じ AWS アカウント の AWS Identity and Access Management (IAM) プリンシパルを使用して S3 オブジェクトをアップロードまたはアクセスする場合は、AWS マネージドキー (aws/s3) を使用できます。
- S3 オブジェクトにクロスアカウントアクセスを許可する場合は、カスタマーマネージドキーを使用します。カスタマーマネージドキーのポリシーを設定して、別のアカウントからのアクセスを許可することができます。
- カスタマーマネージド KMS キーを指定している場合、完全修飾 KMS キー ARN を使用することをお勧めします。代わりに KMS キーエイリアスを使用する場合、AWS KMS はリクエスタのアカウント内でキーを解決します。この動作により、バケット所有者ではなく、リクエスタに属する KMS キーでデータが暗号化される可能性があります。
- お客様 (リクエスタ) が `Encrypt` アクセス許可を付与されているキーを指定する必要があります。詳細については、AWS Key Management Service デベロッパーガイドの「キーユーザーが暗号化オペレーションに KMS キーを使用することを許可する」を参照してください。

カスタマーマネージドキーと AWS マネージド KMS キーをどのような場合に使用するかの詳細については、「Amazon S3 にあるオブジェクトの暗号化に AWS マネージドキー、またはカスタマーマネージドキーを使うべきですか」を参照してください。

SSE-KMS 暗号化ワークフロー

AWS マネージドキー またはカスタマーマネージドキーを使用してデータの暗号化を選択する場合は、AWS KMS および Amazon S3 は次のエンベロープ暗号化アクションを実行します。

1. Amazon S3 は、プレーンテキストのデータキーおよび指定された KMS キーで暗号化されたキーのコピーをリクエストします。

2. AWS KMS は、データキーを生成し、KMS キーで暗号化して、プレーンテキストデータキーと暗号化されたデータキーの両方を Amazon S3 に送信します。
3. Amazon S3 は、データキーを使用してデータを暗号化し、使用後はメモリからプレーンテキストのキーをできる限り迅速に削除します。
4. Amazon S3 は、暗号化されたデータキーを、暗号化されたデータのメタデータとして保存します。

データの復号をリクエストすると、Amazon S3 と AWS KMS は次のアクションを実行します。

1. `Decrypt` リクエストがあると、Amazon S3 は暗号化されたデータキーを AWS KMS に送信します。
2. AWS KMS は、同じ KMS キーを使用して暗号化データキーを復号し、プレーンテキストのデータキーを Amazon S3 に返します。
3. Amazon S3 は、プレーンテキストデータキーを使用して暗号化されたデータを復号し、できるだけ早くプレーンテキストデータキーをメモリから削除します。

> **Important**
>
> Amazon S3 でサーバー側の暗号化に AWS KMS key を使用する場合は、対称暗号化 KMS キーを選択する必要があります。Amazon S3 は、対称暗号化 KMS キーのみをサポートします。このキーの詳細については、「AWS Key Management Service デベロッパーガイド」の「対称暗号化 KMS キー」を参照してください。

SSE-KMS 暗号化の監査

SSE-KMS を指定するリクエストを識別するには、Amazon S3 ストレージレンズメトリクスの「すべての SSE-KMS リクエスト」と「すべての SSE-KMS リクエストの割合 (%)」メトリクスを使用できます。S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージ分析機能です。SSE-KMS 対応バケット数と % SSE-KMS 対応バケットを使用して、デフォルトのバケット暗号化 のバケット数 (SSE-KMS) を把握することもできます。詳細については、「S3 Storage Lens を使用したストレージのアクティビティと使用状況の評価」を参照してください。メトリクスの完全なリストについては、「S3 ストレージレンズメトリクスに関する用語集」を参照してください。

SSE-KMS で暗号化されたデータの AWS KMS キーの使用状況を監査するには、AWS CloudTrail ログを使用します。GenerateDataKey や Decrypt などの暗号化オペレーションに関するインサイトを得ることができます。CloudTrail は、イベント名、ユーザー名、イベントソースなど、検索をフィルタリングするための多数の属性値をサポートしています。

Amazon S3 バケットキー

AWS KMS (SSE-KMS) を使用してサーバー側の暗号化を設定する場合、SSE-KMS で S3 バケットキーを使用するようにバケットを設定できます。SSE-KMS でこのバケットレベルのキーを使用すると、Amazon S3 から AWS KMS へのリクエストトラフィックを減らすことにより、AWS KMS のリクエストコストを最大 99% 削減できます。

新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するようにバケットを設定すると、AWS KMS によりバケットレベルのキーが生成されます。このキーは、バケット内のオブジェクトの一意のデータキーを作成するために使用されます。この S3 バケットキーは Amazon S3 内で期間限定で使用されるため、Amazon S3 で AWS KMS にリクエストを実行し、暗号化オペレーションを完了する必要性が軽減されます。S3 バケットキーの使用の詳細については、Amazon S3 バケットキーを使用した SSE-KMS のコストの削減 を参照してください。

サーバー側の暗号化の要求

特定の Amazon S3 バケット内のすべてのオブジェクトのサーバー側の暗号化を要求するには、バケットポリシーを使用できます。例えば、SSE-KMS を使用したサーバー側の暗号化を要求する `x-amz-server-side-encryption-aws-kms-key-id` ヘッダーがリクエストに含まれていない場合、次のバケットポリシーはすべてのユーザーに対し、オブジェクト (`s3:PutObject`) をアップロードするアクセス許可を拒否します。

```
{
   "Version":"2012-10-17",
   "Id":"PutObjectPolicy",
   "Statement":[{
         "Sid":"DenyObjectsThatAreNotSSEKMS",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::example-s3-bucket1/*",
         "Condition":{
            "Null":{
               "s3:x-amz-server-side-encryption-aws-kms-key-id":"true"
            }
```

```
            }
        }
    ]
}
```

特定の AWS KMS key を使用してバケット内のオブジェクトを暗号化することを要求するには、`s3:x-amz-server-side-encryption-aws-kms-key-id` 条件キーを使用できます。KMS キーを指定するには、キーの Amazon リソースネーム (ARN) を `arn:aws:kms:region:acct-id:key/key-id` 形式で使用する必要があります。AWS Identity and Access Management は、`s3:x-amz-server-side-encryption-aws-kms-key-id` の文字列が存在するかどうかを検証しません。

> **Note**
>
> オブジェクトをアップロードするときに、`x-amz-server-side-encryption-aws-kms-key-id` ヘッダーを使用して KMS キーを指定できます。ヘッダーがリクエストにない場合、Amazon S3 は AWS マネージドキー マネージドキーを使用すると見なします。その場合でも、Amazon S3 がオブジェクトの暗号化に使用する AWS KMS キー ID とポリシーの AWS KMS キー ID は一致する必要があります。一致しない場合、Amazon S3 はリクエストを拒否します。

Amazon S3 固有の条件キーの完全なリストについては、「サービス認証リファレンス」の「Amazon 3 の条件キー」を参照してください。

暗号化コンテキスト

暗号化コンテキストは、データに関する追加のコンテキスト情報が含まれたキーと値のペアのセットです。この暗号化コンテキストは暗号化されていません。暗号化オペレーションで暗号化コンテキストを指定すると、Amazon S3 では同じ暗号化コンテキストを復号オペレーションに指定する必要があります。そうでない場合は、復号に失敗します。AWS KMS は、暗号化コンテキストを追加の認証データ (AAD) として使用し、認証された暗号化をサポートします。暗号化コンテキストの詳細については、「AWS Key Management Service デベロッパーガイド」の「暗号化コンテキスト」を参照してください。

デフォルトでは、Amazon S3 は、オブジェクトまたはバケットの Amazon リソースネーム (ARN) を暗号化コンテキストペアとして使用します。

- S3 バケットキーを有効にせずに SSE-KMS を使用する場合は、オブジェクト ARN を暗号化コンテキストとして使用します。

```
arn:aws:s3:::object_ARN
```

- S3 バケットキーを有効にして SSE-KMS を使用する場合は、暗号化コンテキストとしてバケット ARN を使用します。S3 バケットキーの詳細については、Amazon S3 バケットキーを使用した SSE-KMS のコストの削減 を参照してください。

```
arn:aws:s3:::bucket_ARN
```

s3:PutObject リクエストの `x-amz-server-side-encryption-context` のヘッダーを使って、オプションで追加の暗号化コンテキストペアを指定することができます。しかし、暗号化コンテキストは暗号化されないため、機密情報を含めないでください。Amazon S3 は、この追加のキーペアをデフォルトの暗号化コンテキストとともに保存します。PUT リクエストを処理すると、Amazon S3 は `aws:s3:arn` のデフォルトの暗号化コンテキストを、指定するものに追加します。

暗号化コンテキストを使用して、暗号化オペレーションを識別および分類できます。デフォルトの暗号化コンテキスト ARN 値を使用して、どの Amazon S3 ARN がどの暗号化キーで使用されたかを確認することで、AWS CloudTrail の関連するリクエストを追跡することもできます。

CloudTrail ログファイルの `requestParameters` フィールドでは、暗号化コンテキストは次のようになります。

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::example-s3-bucket1/file_name"
}
```

SSE-KMS をオプションの S3 バケットキー機能とともに使用すると、暗号化コンテキストの値はバケットの ARN になります。

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::example-s3-bucket1"
}
```

AWS KMS 暗号化されたオブジェクトへのリクエストの送信

> **Important**
>
> AWS KMS により暗号化されたオブジェクトに対するすべての GET および PUT リクエストは、Secure Sockets Layer (SSL) または Transport Layer Security (TLS) を使用して生成する必要があります。また、各リクエストは、AWS 署名バージョン 4 (または AWS 署名バージョン 2) のような有効な認証情報を使用して署名する必要もあります。

AWS Signature Version 4 は、HTTP で送信される AWS リクエストに認証情報を追加するプロセスです。セキュリティ対策として、AWS へのほとんどのリクエストは、アクセスキーを使用して署名する必要があります。アクセスキーは、アクセスキー ID とシークレットアクセスキーで構成されます。これらの 2 つのキーは、一般的にセキュリティ認証情報と呼ばれます。詳細については、「リクエストの認証 (AWS 署名バージョン 4)」および「署名バージョン 4 の署名プロセス」を参照してください。

> **Important**
>
> オブジェクトで SSE-KMS を使用している場合、GET リクエストおよび HEAD リクエストに対して暗号化リクエストヘッダーを送信しないでください。そうしないと、HTTP 400 Bad Request エラーが表示されます。

トピック

- AWS KMS (SSE-KMS) によるサーバー側の暗号化の指定
- Amazon S3 バケットキーを使用した SSE-KMS のコストの削減

AWS KMS (SSE-KMS) によるサーバー側の暗号化の指定

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための

自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

すべての Amazon S3 バケットにはデフォルトで暗号化が設定されており、S3 バケットにアップロードされたすべての新しいオブジェクトは保存時に自動的に暗号化されます。Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) は、Amazon S3 のすべてのバケットでのデフォルトの暗号化設定です。別のタイプの暗号化を使用するには、S3 PUT リクエストで使用するサーバー側の暗号化のタイプを指定するか、宛先バケットにデフォルトの暗号化設定を設定できます。

PUT リクエストで別の暗号化タイプを指定する場合は、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS)、または顧客提供のキーによるサーバー側の暗号化 (SSE-C) を使用できます。宛先バケットに別のデフォルト暗号化設定を設定する場合は、SSE-KMS または DSSE-KMS を使用できます。

暗号化は、新しいオブジェクトをアップロードしたり、既存のオブジェクトをコピーしたりするときに適用できます。

Amazon S3 コンソール、REST API オペレーション、AWS SDK、および AWS Command Line Interface (AWS CLI) を使用して SSE-KMS を指定できます。詳細については、以下のトピックを参照してください。

Note

Amazon S3 では、マルチリージョン AWS KMS keys を使用できます。ただし、Amazon S3 では現在、マルチリージョンキーは、単一リージョンキーであるかのように処理され、キーのマルチリージョン特徴は使用しません。詳細については、「AWS Key Management Service デベロッパーガイド」の「マルチリージョンキーを使用する」を参照してください。

Note

別のアカウントが所有している KMS キーを使用する場合は、そのキーを使用するアクセス許可が必要です。KMS キーのクロスアカウント権限の詳細については、「AWS Key

Management Service デベロッパーガイド」の「他のアカウントで使用できる KMS キーを作成する」を参照してください。

S3 コンソールの使用

このトピックでは、Amazon S3 コンソールを使用して、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS) を使用するようにオブジェクトの暗号化のタイプを設定または変更する方法について説明します。

> **Note**
>
> - オブジェクトの暗号化を変更すると、新しいオブジェクトが作成され、古いオブジェクトが置き換えられます。S3 バージョニングが有効になっている場合は、オブジェクトの新しいバージョンが作成され、既存のオブジェクトが古いバージョンになります。また、プロパティを変更するロールは、新しいオブジェクト (またはオブジェクトのバージョン) の所有者になります。
> - ユーザー定義タグを持つオブジェクトの暗号化タイプを変更する場合は、`s3:GetObjectTagging` アクセス許可が必要です。ユーザー定義タグがなく、サイズが 16 MB を超えるオブジェクトの暗号化タイプを変更する場合は、`s3:GetObjectTagging` アクセス許可も必要です。
>
>   ターゲットバケットポリシーによって `s3:GetObjectTagging` アクションが拒否されると、オブジェクトの暗号化タイプは更新されますが、ユーザー定義タグはオブジェクトから削除され、エラーが発生します。

オブジェクトの暗号化を追加または変更するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [Buckets] (バケット) リストで、オブジェクトが含まれるバケットの名前を選択します。
4. [オブジェクト] リストで、暗号化を追加または変更するオブジェクトの名前を選択します。

   オブジェクトの詳細ページが開き、オブジェクトのプロパティを表示するいくつかのセクションが表示されます。

5. [プロパティ] タブを選択します。
6. 「サーバー側の暗号化設定」セクションまでスクロールして、[編集]を選択します。

   [Edit server-side encryption (サーバー側の暗号化を編集する)] ページが開きます。
7. [サーバー側の暗号化] の [暗号化設定] で、[デフォルトの暗号化バケット設定を上書きする] を選択します。
8. [暗号化タイプ] で、[AWS Key Management Service キーによるサーバー側の暗号化 (SSE-KMS)] を選択します。

   > **Important**
   >
   > デフォルト暗号化設定に SSE-KMS オプションを使用する場合、AWS KMS の 1 秒あたりのリクエスト (RPS) 制限が適用されます。AWS KMS クォータの詳細およびクォータの引き上げをリクエストする方法については、「AWS Key Management Service デベロッパーガイド」の「クォータ」を参照してください。
9. [AWS KMS キー] で、次のいずれかを実行して KMS キーを選択します。

   - 使用可能な KMS キーのリストから選択するには、[AWS KMS keys から選択する] を選択し、使用可能なキーのリストから自分の KMS キーを選択します。

     AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がこのリストに表示されます。カスタマーマネージドキーの詳細については、AWS Key Management Serviceデベロッパーガイドの「カスタマーキーと AWS キー」を参照してください。
   - KMS キー ARN を入力するには、[AWS KMS key ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。
   - AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

     AWS KMS key の作成の詳細については、AWS Key Management Service デベロッパーガイドのキーの作成を参照してください。

   > **Important**
   >
   > バケットと同じ AWS リージョン で使用可能な KMS キーのみを使用できます。Amazon S3 コンソールには、バケットと同じリージョンで最初の 100 個の KMS キーしか表示されません。リストに存在しない KMS キーを使用するには、KMS キー ARN を入力する

必要があります。別のアカウントが所有している KMS キーを使用する場合は、まずそのキーを使用するアクセス許可が必要であり、次に KMS キー ARN を入力する必要があります。
Amazon S3 では、対称暗号化 KMS キーのみがサポートされ、非対称暗号化 KMS キーはサポートされません。詳細については、AWS Key Management Service デベロッパーガイドの「Identifying symmetric and asymmetric KMS keys」(対称および非対称 KMS キーの識別) を参照してください。

10. [Save changes] (変更の保存) をクリックします。

> **Note**
>
> このアクションは、指定されたすべてのオブジェクトに暗号化を適用します。フォルダを暗号化する場合は、保存オペレーションが完了するのを待ってから、フォルダに新しいオブジェクトを追加します。

REST API の使用

オブジェクトを作成するとき (新しいオブジェクトをアップロードするか、既存のオブジェクトをコピーするとき) に、AWS KMS keys (SSE-KMS) を使用したサーバー側の暗号化の使用を指定してデータを暗号化できます。これを行うには、リクエストに `x-amz-server-side-encryption` ヘッダーを追加します。ヘッダーの値を、暗号化アルゴリズム `aws:kms` に設定します。Amazon S3 によりレスポンスヘッダー `x-amz-server-side-encryption` が返されるため、SSE-KMS を使用してオブジェクトが保存されたことを確認できます。

`aws:kms` の値で `x-amz-server-side-encryption` ヘッダーを指定する場合は、次のリクエストヘッダーも使用できます。

- `x-amz-server-side-encryption-aws-kms-key-id`
- `x-amz-server-side-encryption-context`
- `x-amz-server-side-encryption-bucket-key-enabled`

トピック

- SSE-KMS をサポートする Amazon S3 REST API オペレーション
- 暗号化コンテキスト (x-amz-server-side-encryption-context)

- AWS KMS キー ID (x-amz-server-side-encryption-aws-kms-key-id)
- S3 バケットキー (x-amz-server-side-encryption-aws-bucket-key-enabled)

SSE-KMS をサポートする Amazon S3 REST API オペレーション

次の REST API オペレーションでは `x-amz-server-side-encryption`、`x-amz-server-side-encryption-aws-kms-key-id`、および `x-amz-server-side-encryption-context` リクエストヘッダーを受け入れます。

- PutObject - PUT オペレーションを使用してデータをアップロードするとき、これらのリクエストヘッダーを指定できます。
- CopyObject — オブジェクトをコピーするときには、ソースオブジェクトとターゲットオブジェクトの両方があります。`CopyObject` オペレーションで SSE-KMS ヘッダーを渡す場合は、ターゲットオブジェクトにのみ適用されます。既存のオブジェクトをコピーする際は、コピー元オブジェクトが暗号化されているかどうかに関係なく、明示的にサーバー側の暗号化を要求しない限り、コピー先オブジェクトは暗号化されません。
- POST Object — POST オペレーションを使用してオブジェクトをアップロードするときには、リクエストヘッダーの代わりに、フォームフィールドで同じ情報を指定します。
- CreateMultipartUpload — マルチパートアップロード API オペレーションを使用して大きいオブジェクトをアップロードする場合、これらのヘッダーを指定できます。これらのヘッダーを `CreateMultipartUpload` リクエストで指定します。

次の REST API オペレーションのレスポンスヘッダーは、オブジェクトがサーバー側の暗号化を使用して保存されているときに `x-amz-server-side-encryption` ヘッダーを返します。

- PutObject
- CopyObject
- POST Object
- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- GetObject
- HeadObject

> **Important**
>
> - AWS KMS で保護されたオブジェクトに対する GET リクエストと PUT リクエストはすべて、Secure Sockets Layer (SSL)、Transport Layer Security (TLS)、または署名バージョン 4 を使用していない場合、失敗します。
> - オブジェクトで SSE-KMS を使用している場合、GET リクエストおよび HEAD リクエストに対して暗号化リクエストヘッダーを送信しないでください。送信すると、HTTP 400 BadRequest エラーが発生します。

暗号化コンテキスト (**x-amz-server-side-encryption-context**)

x-amz-server-side-encryption:aws:kms を指定した場合、Amazon S3 API は x-amz-server-side-encryption-context ヘッダーの暗号化コンテキストをサポートします。暗号化コンテキストは、データに関する追加のコンテキスト情報が含まれたキーバリューペアのセットです。

Amazon S3 は、オブジェクトまたはバケットの Amazon リソースネーム (ARN) を暗号化コンテキストペアとして自動的に使用します。S3 バケットキーを有効にせずに SSE-KMS を使用する場合は、オブジェクト ARN を暗号化コンテキストとして使用します。例えば、arn:aws:s3:::*object_ARN* です。ただし、S3 バケットキーを有効にして SSE-KMS を使用する場合は、バケット ARN を暗号化コンテキストに使用します。例えば、arn:aws:s3:::*bucket_ARN* です。

オプションで、x-amz-server-side-encryption-context ヘッダーを使用して、追加の暗号化コンテキストペアを指定することもできます。しかし、暗号化コンテキストは暗号化されないため、機密情報を含めないでください。Amazon S3 は、この追加のキーペアをデフォルトの暗号化コンテキストとともに保存します。

Amazon S3 の暗号化コンテキストの詳細については、「暗号化コンテキスト」を参照してください。暗号化コンテキストの一般的な情報については、「AWS Key Management Service デベロッパーガイド」の「AWS Key Management Service Concepts - Encryption context」を参照してください。

AWS KMS キー ID (**x-amz-server-side-encryption-aws-kms-key-id**)

x-amz-server-side-encryption-aws-kms-key-id ヘッダーを使用して、データの保護に使用するカスタマーマネージドキーの ID を指定できます。x-amz-server-side-

encryption:aws:kms ヘッダーを指定しても x-amz-server-side-encryption-aws-kms-key-id ヘッダーを指定しない場合、Amazon S3 は AWS マネージドキー (aws/s3) を使用してデータを保護します。カスタマーマネージドキーを使用する場合は、カスタマーマネージドキーの x-amz-server-side-encryption-aws-kms-key-id ヘッダーを指定する必要があります。

> **Important**
>
> Amazon S3 でサーバー側の暗号化に AWS KMS key を使用する場合は、対称暗号化 KMS キーを選択する必要があります。Amazon S3 は、対称暗号化 KMS キーのみをサポートします。このキーの詳細については、「AWS Key Management Service デベロッパーガイド」の「対称暗号化 KMS キー」を参照してください。

S3 バケットキー (**x-amz-server-side-encryption-aws-bucket-key-enabled**)

x-amz-server-side-encryption-aws-bucket-key-enabled リクエストヘッダーを使用して、オブジェクトレベルで S3 バケットキーを有効または無効にできます。S3 バケットキーで、Amazon S3 から AWS KMS へのリクエストトラフィックを減らすことにより、AWS KMS リクエストコストを削減できます。詳細については、「Amazon S3 バケットキーを使用した SSE-KMS のコストの削減」を参照してください。

x-amz-server-side-encryption:aws:kms ヘッダーを指定しても、x-amz-server-side-encryption-aws-bucket-key-enabled は指定しない場合、オブジェクトは送信先バケットの S3 バケットキーの設定を使用して、オブジェクトを暗号化します。詳細については、「オブジェクトレベルで S3 バケットキーを設定する 」を参照してください。

AWS CLI の使用

次の AWS CLI コマンド例を使用する際は、*user input placeholders* をユーザー自身の情報に置き換えます。

新しいオブジェクトをアップロードするか、既存のオブジェクトをコピーするときに、AWS KMS キーを使用したサーバー側の暗号化の使用を指定してデータを暗号化できます。これを行うには、リクエストに --server-side-encryption aws:kms ヘッダーを追加します。--ssekms-key-id *example-key-id*を使用して、作成したカスタマーマネージド AWS KMS キーを追加します。AWS KMS キー ID を指定せずに --server-side-encryption aws:kms を指定した場合、Amazon S3 は AWS マネージドキーを使用します。

```
aws s3api put-object --bucket example-s3-bucket --key example-object-key --server-side-
encryption aws:kms --ssekms-key-id example-key-id --ssekms-encryption-context example-
encryption-context --body filepath
```

--bucket-key-enabled または --no-bucket-key-enabled を追加することで、put-object または copy-object オペレーションで S3 バケットキーを有効または無効にできます。S3 バケットキーで、Amazon S3 から AWS KMS へのリクエストトラフィックを減らすことにより、AWS KMS リクエストコストを削減できます。詳細については、「Reducing the cost of SSE-KMS with S3 Bucket Keys」を参照してください。

```
aws s3api put-object --bucket example-s3-bucket --key example-object-key --server-side-
encryption aws:kms --bucket-key-enabled --body filepath
```

ソースバケットから新しいバケットにオブジェクトをコピーし、SSE-KMS 暗号化を指定できます。

```
aws s3api copy-object --copy-source example-s3-bucket/example-object-key --
bucket example-s3-bucket2 --key example-object-key --server-side-encryption aws:kms --
sse-kms-key-id example-key-id --ssekms-encryption-context example-encryption-context
```

AWS SDK の使用

AWS SDK を使用する場合、サーバー側の暗号化に AWS KMS keys を使用するよう Amazon S3 にリクエストできます。次の例は、AWS SDK for Java および .NET で SSE-KMS を使用する方法を示しています。他の SDK については、AWS デベロッパーセンターのサンプルコードとライブラリを参照してください。

> **Important**
>
> Amazon S3 でサーバー側の暗号化に AWS KMS key を使用する場合は、対称暗号化 KMS キーを選択する必要があります。Amazon S3 は、対称暗号化 KMS キーのみをサポートします。このキーの詳細については、「AWS Key Management Service デベロッパーガイド」の「対称暗号化 KMS キー」を参照してください。

CopyObject オペレーション

オブジェクトをコピーする場合は、同じリクエストプロパティ (ServerSideEncryptionMethod および ServerSideEncryptionKeyManagementServiceKeyId) を追加して、AWS KMS key

を使用するように Amazon S3 に対してリクエストします。オブジェクトのコピーの詳細については、オブジェクトのコピー、移動、名前の変更 を参照してください。

PUT オペレーション

Java

AWS SDK for Java を使用してオブジェクトをアップロードする場合、次のリクエストのように SSEAwsKeyManagementParams プロパティを追加することによって、AWS KMS key を使用するように Amazon S3 に対してリクエストできます。

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
   keyName, file).withSSEAwsKeyManagementParams(new SSEAwsKeyManagementParams());
```

この場合、Amazon S3 は AWS マネージドキー (aws/s3) を使用します。詳細については、「AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用」を参照してください。次の例に示すように、オプションで、対称暗号化 KMS キーを作成し、それをリクエストで指定できます。

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
   keyName, file).withSSEAwsKeyManagementParams(new
 SSEAwsKeyManagementParams(keyID));
```

カスタマーマネージドキーの作成の詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMS API のプログラミング」を参照してください。

オブジェクトをアップロードするためのコード例については、以下のトピックを参照してください。これらの例を使用するには、コード例を更新し、前出のコード例に示されているように暗号化情報を提供する必要があります。

- 1 つのオペレーションでオブジェクトをアップロードする場合は、オブジェクトのアップロード を参照してください。
- 高レベルまたは低レベルのマルチパートアップロード API オペレーションを使用するマルチパートアップロードについては、「マルチパートアップロードを使用したオブジェクトのアップロード」を参照してください。

.NET

AWS SDK for .NET を使用してオブジェクトをアップロードする場合、次のリクエストのように ServerSideEncryptionMethod プロパティを追加することによって、AWS KMS key を使用するように Amazon S3 に対してリクエストできます。

```
PutObjectRequest putRequest = new PutObjectRequest
 {
     BucketName = example-s3-bucket,
     Key = keyName,
     // other properties
     ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS
 };
```

この場合、Amazon S3 は AWS マネージドキー を使用します。詳細については、「AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用」を参照してください。次の例に示すように、オプションで、独自の対称暗号化カスタマーマネージドキーを作成し、それをリクエストで指定できます。

```
PutObjectRequest putRequest1 = new PutObjectRequest
{
  BucketName = example-s3-bucket,
  Key = keyName,
  // other properties
  ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS,
  ServerSideEncryptionKeyManagementServiceKeyId = keyId
};
```

カスタマーマネージドキーの作成の詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMS API のプログラミング」を参照してください。

オブジェクトをアップロードするためのコード例については、以下のトピックを参照してください。これらの例を使用するには、コード例を更新し、前出のコード例に示されているように暗号化情報を提供する必要があります。

- 1 つのオペレーションでオブジェクトをアップロードする場合は、オブジェクトのアップロード を参照してください。
- 高レベルまたは低レベルのマルチパートアップロード API オペレーションを使用するマルチパートアップロードについては、「マルチパートアップロードを使用したオブジェクトのアップロード」を参照してください。

署名付き URL

Java

次の例に示すように、AWS KMS key で暗号化されたオブジェクトの署名付き URL を作成する際には、署名バージョン 4 を明示的に指定する必要があります。

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setSignerOverride("AWSS3V4SignerType");
AmazonS3Client s3client = new AmazonS3Client(
        new ProfileCredentialsProvider(), clientConfiguration);
...
```

コード例については、署名付き URL を使用したオブジェクトの共有 を参照してください。

.NET

次の例に示すように、AWS KMS key で暗号化されたオブジェクトの署名付き URL を作成する際には、署名バージョン 4 を明示的に指定する必要があります。

```
AWSConfigs.S3Config.UseSignatureVersion4 = true;
```

コード例については、署名付き URL を使用したオブジェクトの共有 を参照してください。

Amazon S3 バケットキーを使用した SSE-KMS のコストの削減

Amazon S3 バケットキーは、AWS Key Management Service (AWS KMS) を使用した Amazon S3 サーバー側の暗号化のコストを削減します。SSE-KMS でこのバケットレベルのキーを使用すると、Amazon S3 から AWS KMS へのリクエストトラフィックを減らすことにより、AWS KMS のリクエストコストを最大 99% 削減できます。AWS Management Console を数回クリックするだけで、クライアントアプリケーションの変更はせずにバケットを設定し、新しいオブジェクトで SSE-KMS の暗号化に S3 バケットキーを使用することができます。

> **Note**
>
> S3 バケットキーは、AWS Key Management Service (AWS KMS) キーによる二層式サーバー側の暗号化 (DSSE-KMS) ではサポートされていません。

SSE-KMS の S3 バケットキー

SSE-KMS で暗号化された数百万、数十億のオブジェクトにアクセスするワークロードは、AWS KMS に大量のリクエストを生成します。S3 バケットキーなしで SSE-KMS を使用してデータを保護する場合、Amazon S3 はすべてのオブジェクトに対して個々の AWS KMS データキーを使用します。この場合、KMS で暗号化されたオブジェクトに対してリクエストが実行されるたびに、Amazon S3 は AWS KMS を呼び出します。SSE-KMS の仕組みについては、AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用 を参照してください。

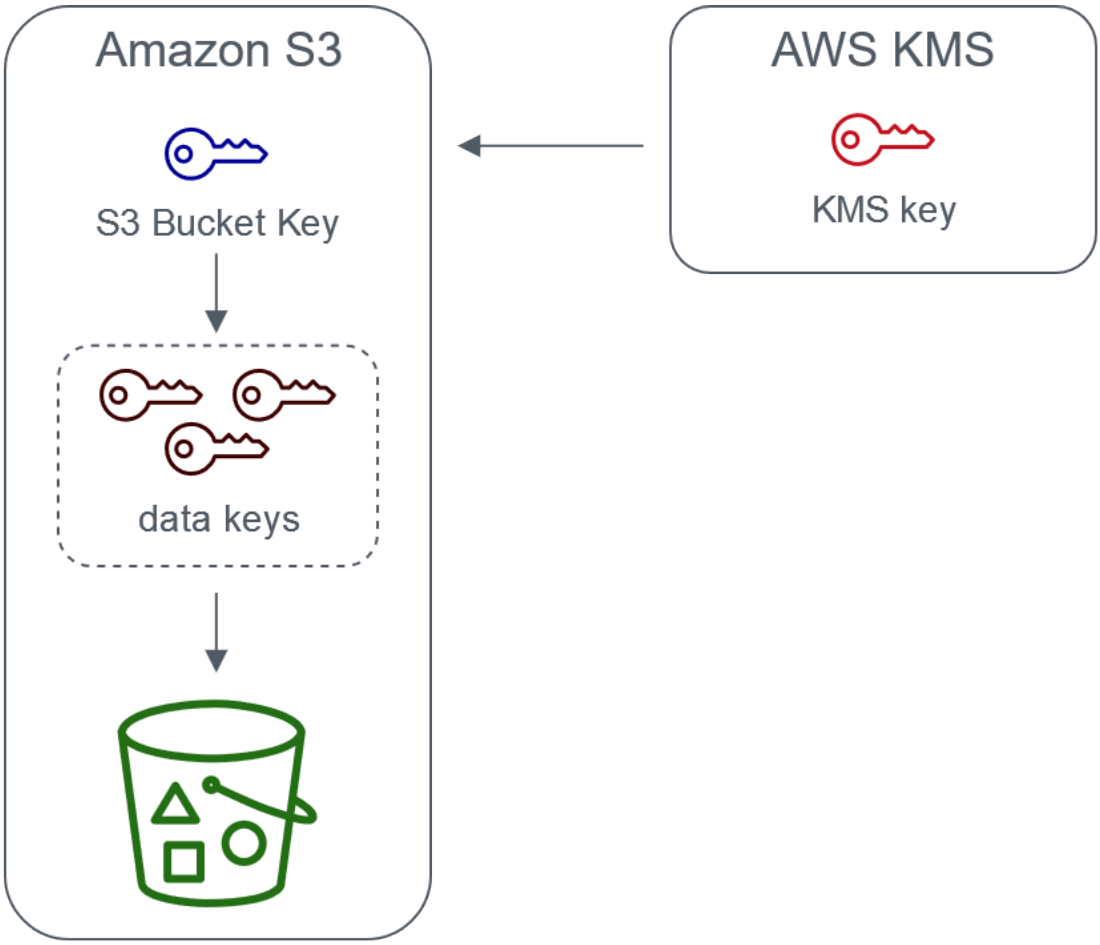
SSE-KMS の S3 バケットキーを使用するようにバケットを設定すると、AWS は AWS KMS から有効期限の短いバケットレベルのキーを生成し、S3 に一時的に保存します。このバケットレベルのキーは、ライフサイクル中に新しいオブジェクトのデータキーを作成します。S3 バケットキーは Amazon S3 内で期間限定で使用されるため、S3 で AWS KMS にリクエストを実行し、暗号化オペレーションを完了する必要性が軽減されます。これにより S3 から AWS KMS へのトラフィックが減少し、これまでと比較してわずかなコストで、Amazon S3 の AWS KMS で暗号化されたオブジェクトにアクセスできるようになります。

固有のバケットレベルのキーはリクエスタごとに少なくとも 1 回フェッチされ、キーへのリクエスターのアクセスが AWS KMS CloudTrail イベントで確実にキャプチャされます。Amazon S3 は、呼び出し元が異なるロールまたはアカウントを使用する場合、または同じロールを異なるスコープポリシーで使用する場合、呼び出し元を異なるリクエスタとして扱います。AWS KMS リクエスト削減数は、リクエスタ数、リクエストパターン、リクエストされたオブジェクトの相対的な経過時間を反映しています。例えば、リクエスタの数が少なく、限られた時間枠で複数のオブジェクトをリクエストして、同じバケットレベルのキーで暗号化した場合、大幅な削減につながります。

> **Note**
>
> S3 バケットキーを使用すると、バケットレベルのキーを使って `Encrypt`、`GenerateDataKey`、`Decrypt` オペレーションの AWS KMS リクエスト数を減らし、AWS KMS リクエストのコストを節約できます。設計上、このバケットレベルのキーを利用する後続のリクエストが AWS KMS API リクエストになったり、AWS KMS キーポリシーに照らしてアクセスが検証されたりすることはありません。

S3 バケットキーを設定する場合、バケット内に既に存在するオブジェクトは S3 バケットキーを使用しません。既存のオブジェクトの S3 バケットキーを設定するには、`CopyObject` オペレーションを使用できます。詳細については、「オブジェクトレベルで S3 バケットキーを設定する 」を参照してください。

Amazon S3 は、同じ AWS KMS key で暗号化されるオブジェクトの S3 バケットキーのみを共有します。S3 バケットキーは、AWS KMS によって作成された KMS キー、インポートされたキーマテリアル、およびカスタムキーストア によってバックアップされたキーマテリアルと互換性があります。



Server-side encryption with AWS Key Management service using an S3 Bucket Key

S3 バケットキーの設定

Amazon S3 コンソール、AWS SDK、AWS CLI、または REST API を使用して、新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するようにバケットを設定できます。バケットで S3 バケットキーを有効にすると、別の指定した SSE-KMS キーでアップロードされたオブジェクトは、独自の S3 バケットキーを使用します。S3 バケットキーの設定にかかわらず、リクエストに `true` または `false` の値の付いた `x-amz-server-side-encryption-bucket-key-enabled` ヘッダーを含めると、バケット設定を上書きできます。

S3 バケットキーを使用するようにバケットを設定する前に、「S3 バケットキーを有効にする前に注意するべき変更点」を確認してください。

Amazon S3 コンソールを使用した S3 バケットキーの設定

新しいバケットを作成する際、新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するようにバケットを設定できます。また、バケットプロパティを更新して、新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するように既存のバケットを設定することもできます。

詳細については、「新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するようにバケットを設定する」を参照してください。

S3 バケットキーに対する REST API、AWS CLI、および AWS SDK のサポート

REST API、AWS CLI、または AWS SDK を使用して、新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するようにバケットを設定できます。オブジェクトレベルで S3 バケットキーを有効にすることもできます。

詳細については、次を参照してください:

- オブジェクトレベルで S3 バケットキーを設定する
- 新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するようにバケットを設定する

次の API オペレーションは、SSE-KMS の S3 バケットキーをサポートしています。

- PutBucketEncryption
  - `ServerSideEncryptionRule` は、S3 バケットキーを有効または無効にする `BucketKeyEnabled` パラメータを受け入れます。
- GetBucketEncryption
  - `ServerSideEncryptionRule` は の設定を返します。`BucketKeyEnabled`
- PutObject、CopyObject、CreateMultipartUpload、および POST Object
  - `x-amz-server-side-encryption-bucket-key-enabled` リクエストヘッダーは、オブジェクトレベルで S3 バケットキーを有効または無効にします。
- HeadObject、GetObject、UploadPartCopy、UploadPart、CompleteMultipartUpload
  - `x-amz-server-side-encryption-bucket-key-enabled` レスポンスヘッダーは、オブジェクトに対して S3 バケットキーが有効か無効かを示します。

AWS CloudFormation の使用

AWS CloudFormation では、`AWS::S3::Bucket` のリソースに S3 バケットキーを有効または無効にするために使用できる `BucketKeyEnabled` と呼ばれる暗号化プロパティが含まれています。

詳細については、AWS CloudFormation の使用 を参照してください。

S3 バケットキーを有効にする前に注意するべき変更点

S3 バケットキーを有効にする前に、次の関連する変更点に注意してください。

IAM または AWS KMS キーポリシー

既存の AWS Identity and Access Management (IAM) ポリシーまたは AWS KMS キーポリシーで、オブジェクトの Amazon リソースネーム (ARN) を暗号化コンテキストとして使用し、KMS キーへのアクセスを絞り込んだり制限したりする場合、S3 バケットキーではこれらのポリシーは機能しません。S3 バケットキーは、バケット ARN を暗号化コンテキストとして使用します。S3 バケットキーを有効にする前に、バケット ARN を暗号化コンテキストとして使用するため、IAM ポリシーまたは AWS KMS キーポリシーを更新してください。

暗号化コンテキストと S3 バケットキーの詳細については、「暗号化コンテキスト」を参照してください。

AWS KMS の CloudTrail イベント

S3 バケットキーを有効にすると、オブジェクト ARN の代わりに、AWS KMS CloudTrail イベントがバケット ARN を口グに記録します。さらに、口グに表示される SSE-KMS オブジェクトの KMS CloudTrail イベントが少なくなります。これは、Amazon S3 ではキーマテリアルに時間制限があり、これにより AWS KMS に対するリクエストが減るためです。

レプリケーションでの S3 バケットキーの使用

S3 バケットキーは、同一リージョンレプリケーション (SRR) およびクロスリージョンレプリケーション (CRR) で使用できます。

Amazon S3 では、暗号化されたオブジェクトをレプリケートする場合、通常、レプリケート先バケット内のレプリカオブジェクトの暗号化設定が保持されます。ただし、ソースオブジェクトが暗号化されておらず、レプリケート先バケットがデフォルトの暗号化または S3 バケットキーを使用している場合、Amazon S3 はレプリケート先のバケットの設定でオブジェクトを暗号化します。

以下の例では、S3 バケットキーがレプリケーションでどのように機能するかを示しています。詳細については、「暗号化されたオブジェクトのレプリケート (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)」を参照してください。

Example 例 1 – ソースオブジェクトで S3 バケットキーを使用し、レプリケート先バケットでデフォルトの暗号化を使用する

ソースオブジェクトでは S3 バケットキーを使用しているが、レプリケート先バケットでは SSE-KMS によるデフォルトの暗号化を使用している場合、レプリカオブジェクトは S3 バケットキーの暗号化設定をレプリケート先バケットに保持します。レプリケート先バケットでは、SSE-KMS によるデフォルトの暗号化が使用されます。

Example 例 2 – ソースオブジェクトが暗号化されておらず、レプリケート先バケットで SSE-KMS の S3 バケットキーを使用する

ソースオブジェクトが暗号化されておらず、レプリケート先のバケットで SSE-KMS の S3 バケットキーを使用する場合、レプリカオブジェクトはレプリケート先バケットで SSE-KMS を使用して、S3 バケットキーで暗号化されます。この結果、ソースオブジェクトの ETag はレプリカオブジェクトの ETag と異なることになります。この違いに対応するには、ETag を使用するアプリケーションを更新する必要があります。

S3 バケットキーの使用

S3 バケットキーの有効化と使用の詳細については、以下のセクションを参照してください。

- 新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するようにバケットを設定する
- オブジェクトレベルで S3 バケットキーを設定する
- S3 バケットキーの設定の表示

新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するようにバケットを設定する

AWS Key Management Service (AWS KMS) キー (SSE-KMS) を使用してサーバー側の暗号化を設定する場合、新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するようにバケットを設定できます。S3 バケットキーは、Amazon S3 から AWS KMS へのリクエストトラフィックを減らし、SSE-KMS におけるコストを削減します。詳細については、「Amazon S3 バケットキーを使用した SSE-KMS のコストの削減」を参照してください。

Amazon S3 コンソール、REST API、AWS SDK、AWS Command Line Interface (AWS CLI)、または AWS CloudFormation を使用して、新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するようにバケットを設定できます。既存のオブジェクトの S3 バケットキーを有効または無効にするには、CopyObject オペレーションを使用できます。詳細については、(オブジェクトレベルで S3 バケットキーを設定する) および (S3 バッチオペレーションを使用した S3 バケットキーによるオブジェクトの暗号化) を参照してください。

レプリケート元バケットまたはレプリケート先バケットで S3 バケットキーを有効にすると、暗号化コンテキストはバケットの Amazon リソースネーム (ARN) になり、オブジェクト ARN にはなりません (例えば、`arn:aws:s3:::bucket_ARN`)。IAM ポリシーを更新して、暗号化コンテキストにバケット ARN を使用する必要があります。詳細については、「S3 バケットキーとレプリケーション」を参照してください。

以下の例では、S3 バケットキーがレプリケーションでどのように機能するかを示しています。詳細については、「暗号化されたオブジェクトのレプリケート (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)」を参照してください。

前提条件

S3 バケットキーを使用するようにバケットを設定する前に、「S3 バケットキーを有効にする前に注意するべき変更点」を確認してください。

S3 コンソールの使用

S3 コンソールでは、新しいバケットまたは既存のバケットに対して S3 バケットキーを有効または無効にできます。S3 コンソールのオブジェクトは、バケット設定からそれらの S3 バケットキーの設定を継承します。バケットで S3 バケットキーを有効にすると、バケットにアップロードする新しいオブジェクトでは、SSE-KMS に S3 バケット キーが使用されます。

S3 バケットキーが有効になっているバケット内のオブジェクトをアップロード、コピー、または変更する

S3 バケットキーが有効になっているバケットでオブジェクトをアップロード、変更、またはコピーすると、そのオブジェクトの S3 バケットキーの設定がバケット設定に合わせて更新されることがあります。

オブジェクトで既に S3 バケットキーが有効になっている場合、オブジェクトをコピーまたは変更しても、そのオブジェクトの S3 バケットキーの設定は変更されません。ただし、レプリケート先バケットに S3 バケットキーの設定がある状態で S3 バケットキーが有効になっていないオブジェクトを変更またはコピーした場合、オブジェクトはレプリケート先バケットの S3 バケットキーの設定を継承します。例えば、ソースオブジェクトで S3 バケットキーが有効になっていないが、レプリケート先バケットで S3 バケットキーが有効になっている場合、S3 バケットキーは有効になります。

新しいバケットの作成時に S3 バケットキーを有効にするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。

3. [Create bucket (バケットの作成)] を選択します。
4. バケット名を入力し、AWS リージョンを選択します。
5. [デフォルトの暗号化] の [暗号化キーの種類] で、[AWS Key Management Service キー (SSE-KMS)] を選択します。
6. [AWS KMS キー] で、次のいずれかを実行して KMS キーを選択します。

   - 使用可能な KMS キーのリストから選択するには、[AWS KMS keys から選択する] を選択し、使用可能なキーのリストから自分の KMS キーを選択します。

     AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がこのリストに表示されます。カスタマーマネージドキーの詳細については、AWS Key Management Serviceデベロッパーガイドの「カスタマーキーと AWS キー」を参照してください。
   - KMS キー ARN を入力するには、[AWS KMS key ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。
   - AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

     AWS KMS key の作成の詳細については、「AWS Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。
7. [バケットキー] で [有効化] を選択します。
8. [Create bucket (バケットの作成)] を選択します。

   Amazon S3 は、S3 バケットキーを有効にしてバケットを作成します。バケットにアップロードする新しいオブジェクトには、S3 バケットキーを使用します。

   S3 バケットキーを無効にするには、前の手順に従い、[Disable (無効化)] を選択します。

既存のバケットに対して S3 バケットキーを有効にするには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、S3 バケットキーを有効にするバケットを選択します。
4. [プロパティ] タブを選択します。
5. [デフォルトの暗号化] で、[編集] を選択します。
6. [デフォルトの暗号化] の [暗号化キーの種類] で、[AWS Key Management Service キー (SSE-KMS)] を選択します。

7. [AWS KMS キー] で、次のいずれかを実行して KMS キーを選択します。

   - 使用可能な KMS キーのリストから選択するには、[AWS KMS keys から選択する] を選択し、使用可能なキーのリストから自分の KMS キーを選択します。

     AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がこのリストに表示されます。カスタマーマネージドキーの詳細については、AWS Key Management Serviceデベロッパーガイドの「カスタマーキーと AWS キー」を参照してください。
   - KMS キー ARN を入力するには、[AWS KMS key ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。
   - AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

     AWS KMS key の作成の詳細については、「AWS Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。
8. [バケットキー] で [有効化] を選択します。
9. [Save changes] (変更の保存) をクリックします。

   Amazon S3 では、バケットに追加された新しいオブジェクトに対して S3 バケットキーが有効になります。既存のオブジェクトでは S3 バケットキーは使用しません。既存のオブジェクトの S3 バケットキーを設定するには、CopyObject オペレーションを使用できます。詳細については、「オブジェクトレベルで S3 バケットキーを設定する 」を参照してください。

   S3 バケットキーを無効にするには、前の手順に従い、[Disable (無効化)] を選択します。

REST API の使用

PutBucketEncryption を使用してバケットの S3 バケットキーを有効または無効にできます。PutBucketEncryption で S3 バケットキーを設定するには、ServerSideEncryptionRule を使用します。これには、SSE-KMS を使用したデフォルトの暗号化が含まれます。また、オプションで、カスタマーマネージドキーの KMS キー ID を指定して、カスタマーマネージドキーを使用することもできます。

詳細と構文の例については、PutBucketEncryption を参照してください。

AWS SDK for Java の使用

次の例では、AWS SDK for Java を使用して、S3 バケットキーと SSE-KMS によるデフォルトのバケット暗号化を有効にします。

Java

```
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .build();

ServerSideEncryptionByDefault serverSideEncryptionByDefault = new
 ServerSideEncryptionByDefault()
    .withSSEAlgorithm(SSEAlgorithm.KMS);
ServerSideEncryptionRule rule = new ServerSideEncryptionRule()
    .withApplyServerSideEncryptionByDefault(serverSideEncryptionByDefault)
    .withBucketKeyEnabled(true);
ServerSideEncryptionConfiguration serverSideEncryptionConfiguration =
    new ServerSideEncryptionConfiguration().withRules(Collections.singleton(rule));

SetBucketEncryptionRequest setBucketEncryptionRequest = new
 SetBucketEncryptionRequest()
    .withServerSideEncryptionConfiguration(serverSideEncryptionConfiguration)
    .withBucketName(bucketName);

s3client.setBucketEncryption(setBucketEncryptionRequest);
```

AWS CLI の使用

次の例では、AWS CLI を使用して、S3 バケットキーと SSE-KMS によるデフォルトのバケット暗号化を有効にします。*user input placeholders* を、ユーザー自身の情報に置き換えます。

```
aws s3api put-bucket-encryption --bucket example-s3-bucket --server-side-encryption-
configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

AWS CloudFormation の使用

AWS CloudFormation を使用して S3 バケットキーを設定する方法の詳細については、AWS CloudFormation ユーザーガイドの「AWS::S3::Bucket ServerSideEncryptionRule」を参照してください。

オブジェクトレベルで S3 バケットキーを設定する

REST API、AWS SDK、または AWS CLI を使用して PUT オペレーションまたは COPY オペレーションを実行する際、`true` または `false` の値の付いた `x-amz-server-side-encryption-bucket-key-enabled` リスエストヘッダーを追加して、オブジェクトレベルで S3 バケットキーを有効または無効にできます。S3 バケットキーは、Amazon S3 から AWS KMS へのリクエストトラフィックを減らすことにより、AWS Key Management Service (AWS KMS) (SSE-KMS) を使用したサーバー側の暗号化のコストを削減します。詳細については、「Amazon S3 バケットキーを使用した SSE-KMS のコストの削減」を参照してください。

PUT オペレーションまたは COPY オペレーションを使用してオブジェクトの S3 バケットキーを設定すると、Amazon S3 では、そのオブジェクトの設定のみが更新されます。レプリケート先バケットの S3 バケットキーの設定は変更されません。KMS で暗号化されたオブジェクトの PUT または COPY リクエストを S3 バケットキーが有効になっているバケットに送信すると、リクエストヘッダーのキーを無効にしない限り、オブジェクトレベルのオペレーションでは自動的に S3 バケットキーが使用されます。オブジェクトに S3 バケットキーを指定しない場合、Amazon S3 ではレプリケート先バケットの S3 バケットキーの設定がオブジェクトに適用されます。

前提条件:

S3 バケットキーを使用するようにオブジェクトを設定する前に、「S3 バケットキーを有効にする前に注意するべき変更点」を確認してください。

トピック

- Amazon S3 バッチオペレーション
- REST API の使用
- AWS SDK Java (PutObject) の使用
- AWS CLI (PutObject) の使用

Amazon S3 バッチオペレーション

既存の Amazon S3 オブジェクトを暗号化するには、Amazon S3 バッチオペレーションを使用します。S3 バッチオペレーションは、操作するオブジェクトのリストとともに提供します。バッチオペレーションは各 API を呼び出して、指定されたオペレーションを実行します。

S3 バッチオペレーションのコピーオペレーションを使用して、既存の暗号化されていないオブジェクトをコピーし、同じバケットに新しい暗号化されたオブジェクトを書き込めます。1 つのバッチオペレーションジョブで、数十億ものオブジェクトに対して、指定されたオペレーションを実行できます。詳細については、Amazon S3 オブジェクトでの大規模なバッチ操作の実行 および Amazon S3 バッチオペレーションによるオブジェクトの暗号化を参照してください。

REST API の使用

SSE-KMS を使用すると、次の API を使用して、オブジェクトの S3 バケットキーを有効にできます。

- PutObject – オブジェクトをアップロードするときに、`x-amz-server-side-encryption-bucket-key-enabled` リクエストヘッダーを指定して、オブジェクトレベルで S3 バケットキーを有効または無効にできます。
- CopyObject – オブジェクトをコピーして SSE-KMS を設定するときに、`x-amz-server-side-encryption-bucket-key-enabled` リクエストヘッダーを指定して、オブジェクトの S3 バケットキーを有効または無効にできます。
- PostObject – POST オペレーションを使用してオブジェクトをアップロードし、SSE-KMS を設定するとき、`x-amz-server-side-encryption-bucket-key-enabled` フォームフィールドを使用してオブジェクトの S3 バケットキーを有効または無効にできます。
- CreateMultipartUpload – `CreateMultipartUpload` API オペレーションを使用して大きなオブジェクトをアップロードし、SSE-KMS を設定する場合、`x-amz-server-side-encryption-bucket-key-enabled` リクエストヘッダーを使用してオブジェクトの S3 バケットキーを有効または無効にできます。

オブジェクトレベルで S3 バケットキーを有効にするには、`x-amz-server-side-encryption-bucket-key-enabled` リクエストヘッダーを含めます。SSE-KMS および REST API の詳細については、REST API の使用 を参照してください。

AWS SDK Java (PutObject) の使用

次の例を使用して、AWS SDK for Java を使ってオブジェクトレベルで S3 バケットキーを設定できます。

Java

```
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .build();

String bucketName = "DOC-EXAMPLE-BUCKET1";
String keyName = "key name for object";
String contents = "file contents";

PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, keyName,
 contents)
    .withBucketKeyEnabled(true);

s3client.putObject(putObjectRequest);
```

AWS CLI (PutObject) の使用

次の AWS CLI の例を使用して、PutObject リクエストの一部としてオブジェクトレベルで S3 バケットキーを設定できます。

```
aws s3api put-object --bucket example-s3-bucket --key object key name --server-side-
encryption aws:kms --bucket-key-enabled --body filepath
```

S3 バケットキーの設定の表示

Amazon S3 コンソール、REST API、AWS Command Line Interface (AWS CLI)、または AWS SDK を使用して、バケットまたはオブジェクトレベルで S3 バケットキーの設定を表示できます。

S3 バケットキーは、Amazon S3 から AWS KMS へのリクエストトラフィックを減らし、AWS Key Management Service (SSE-KMS) を使用したサーバー側の暗号化のコストを削減します。詳細については、「Amazon S3 バケットキーを使用した SSE-KMS のコストの削減」を参照してください。

バケット設定から S3 バケット キー設定を継承したバケットまたはオブジェクトの S3 バケット キー設定を表示するには、s3:GetEncryptionConfiguration アクションを実行するアクセス許可が必要です。詳細については、Amazon Simple Storage Service API リファレンスの GetBucketEncryption を参照してください。

S3 コンソールの使用

S3 コンソールでは、バケットまたはオブジェクトの S3 バケットキーの設定を表示できます。ソースオブジェクトにすでに S3 バケットキーが設定されていない限り、S3 バケットキーの設定はバケット設定から継承されます。

同じバケット内のオブジェクトとフォルダは、異なる S3 バケットキーの設定を行えます。例えば、REST API を使用してオブジェクトをアップロードし、そのオブジェクトの S3 バケットキーを有効にした場合、レプリケート先バケットで S3 バケットキーが無効になっている場合でも、S3 バケットキーの設定は保持されます。別の例として、既存のバケットに対して S3 バケットキーを有効にした場合、バケット内に既に存在するオブジェクトには S3 バケットキーを使用できません。ただし、新しいオブジェクトでは S3 バケットキーが有効になります。

バケットの S3 バケットキーの設定を表示するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、S3 バケットキーを有効にするバケットを選択します。
4. [プロパティ] を選択します。
5. [Default encryption (デフォルトの暗号化)] セクションの [バケットキー] に、バケットの S3 バケットキーの設定が表示されます。

   S3 バケットキーの設定が表示されない場合は、`s3:GetEncryptionConfiguration` アクションを実行するアクセス許可がない可能性があります。詳細については、Amazon Simple Storage Service API リファレンスの GetBucketEncryption を参照してください。

オブジェクトの S3 バケットキーの設定を表示するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、S3 バケットキーを有効にするバケットを選択します。
3. [オブジェクト] リストで、オブジェクト名を選択します。
4. [詳細] タブの [Server-side encryption settings (サーバー側の暗号化設定)] で、[Edit (編集)] を選択します。

[バケットキー] の下に、オブジェクトの S3 バケットキーの設定が表示されます。この設定を編集することはできません。

AWS CLI の使用

バケットレベルの S3 バケットキーの設定を返すには

この例を実行するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。

```
aws s3api get-bucket-encryption --bucket example-s3-bucket1
```

詳細については、AWS CLI コマンドリファレンスの「get-bucket-encryption」を参照してください。

S3 バケットキーのオブジェクトレベルの設定を返すには

この例を実行するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。

```
aws s3api head-object --bucket example-s3-bucket1 --key my_images.tar.bz2
```

詳細については、AWS CLI コマンドリファレンスの「head-object」を参照してください。

REST API の使用

バケットレベルの S3 バケットキーの設定を返すには

S3 バケットキーの設定を含むバケットの暗号化情報を返すには、GetBucketEncryption オペレーションを使用します。S3 バケットキーの設定は、BucketKeyEnabled の設定と共に ServerSideEncryptionConfiguration 要素のレスポンス本文に返されます。詳細については、Amazon S3 API リファレンスの GetBucketEncryption を参照してください。

S3 バケットキーのオブジェクトレベルの設定を返すには

オブジェクトの S3 バケットキーのステータスを返すには、HeadObject オペレーションを使用します。HeadObject は、オブジェクトに対して S3 バケットキーが有効か無効かを示す x-amz-server-side-encryption-bucket-key-enabled レスポンスヘッダーを返します。詳細については、Amazon S3 API リファレンス の HeadObject を参照してください。

次の API オペレーションは、オブジェクトに対して S3 バケットキーが設定されている場合、x-amz-server-side-encryption-bucket-key-enabled レスポンスヘッダーも返します。

- PutObject
- PostObject
- CopyObject
- CreateMultipartUpload
- UploadPartCopy
- UploadPart
- CompleteMultipartUpload
- GetObject

AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS) の使用

AWS Key Management Service (AWS KMS) キーによる二層式サーバー側の暗号化 (DSSE-KMS) を使用すると、オブジェクトが Amazon S3 にアップロードされるときに 2 つの暗号化レイヤーが適用されます。DSSE-KMS を使用すると、データに多層暗号化を適用し、暗号化キーを完全に制御する必要があるコンプライアンス基準をより簡単に満たすことができます。

S3 バケットで DSSE-KMS を使用する場合、AWS KMS キーはバケットと同じリージョンに存在する必要があります。また、オブジェクトに対して DSSE-KMS がリクエストされると、オブジェクトのメタデータの一部である S3 チェックサムが暗号化された形式で保存されます。チェックサムの詳細については、「オブジェクトの整合性をチェックする」を参照してください。

DSSE-KMS と AWS KMS keys の使用には、追加料金がかかります。DSSE-KMS 料金の詳細については、『AWS Key Management Service デベロッパーガイド』の「AWS KMS key の概念」と「AWS KMS の料金」を参照してください。

> **Note**
>
> S3 バケットキーは DSSE-KMS ではサポートされていません。

AWS KMS keys キーによる二層式サーバー側の暗号化 (DSSE-KMS) の要求

特定の Amazon S3 バケット内のすべてのオブジェクトの二層式サーバー側の暗号化を要求するには、バケットポリシーを使用できます。例えば、DSSE-KMS を使用したサーバー側の暗号化を要求する `x-amz-server-side-encryption` ヘッダーがリクエストに含まれていない場合、次のバケットポリシーはすべてのユーザーに対し、オブジェクト (`s3:PutObject`) をアップロードするアクセス許可を拒否します。

```
{
            "Version":"2012-10-17",
            "Id":"PutObjectPolicy",
            "Statement":[{
                  "Sid":"DenyUnEncryptedObjectUploads",
                  "Effect":"Deny",
                  "Principal":"*",
                  "Action":"s3:PutObject",
                  "Resource":"arn:aws:s3:::example-s3-bucket1/*",
                  "Condition":{
                     "StringNotEquals":{
                        "s3:x-amz-server-side-encryption":"aws:kms:dsse"
                     }
                  }
               }
            ]
         }
```

トピック

- AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS) の指定

AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS) の指定

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

すべての Amazon S3 バケットにはデフォルトで暗号化が設定されており、S3 バケットにアップロードされたすべての新しいオブジェクトは保存時に自動的に暗号化されます。Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) は、Amazon S3 のすべてのバケットでのデフォルト

の暗号化設定です。別のタイプの暗号化を使用するには、S3 PUT リクエストで使用するサーバー側の暗号化のタイプを指定するか、宛先バケットにデフォルトの暗号化設定を設定できます。

PUT リクエストで別の暗号化タイプを指定する場合は、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS)、または顧客提供のキーによるサーバー側の暗号化 (SSE-C) を使用できます。宛先バケットに別のデフォルト暗号化設定を設定する場合は、SSE-KMS または DSSE-KMS を使用できます。

暗号化は、新しいオブジェクトをアップロードしたり、既存のオブジェクトをコピーしたりするときに適用できます。

Amazon S3 コンソール、Amazon S3 REST API、および AWS Command Line Interface (AWS CLI) を使用して、DSSE-KMS を指定できます。詳細については、以下のトピックを参照してください。

> **Note**
>
> Amazon S3 では、マルチリージョン AWS KMS keys を使用できます。ただし、Amazon S3 では現在、マルチリージョンキーは、単一リージョンキーであるかのように処理され、キーのマルチリージョン特徴は使用しません。詳細については、「AWS Key Management Service デベロッパーガイド」の「マルチリージョンキーを使用する」を参照してください。

> **Note**
>
> 別のアカウントが所有している KMS キーを使用する場合は、そのキーを使用するアクセス許可が必要です。KMS キーのクロスアカウント権限の詳細については、「AWS Key Management Service デベロッパーガイド」の「他のアカウントで使用できる KMS キーを作成する」を参照してください。

S3 コンソールの使用

このセクションでは、Amazon S3 コンソールを使用して、AWS Key Management Service (AWS KMS) キーによる二層式サーバー側の暗号化 (DSSE-KMS) を使用するようにオブジェクトの暗号化のタイプを設定または変更する方法について説明します。

Note

- オブジェクトの暗号化方法を変更すると、新しいオブジェクトが作成され、古いオブジェクトが置き換えられます。S3 バージョニングが有効になっている場合は、オブジェクトの新しいバージョンが作成され、既存のオブジェクトが古いバージョンになります。また、プロパティを変更するロールは、新しいオブジェクト (またはオブジェクトのバージョン) の所有者になります。
- ユーザー定義タグを持つオブジェクトの暗号化タイプを変更する場合は、`s3:GetObjectTagging` アクセス許可が必要です。ユーザー定義タグがなく、サイズが 16 MB を超えるオブジェクトの暗号化タイプを変更する場合は、`s3:GetObjectTagging` アクセス許可も必要です。

  ターゲットバケットポリシーによって `s3:GetObjectTagging` アクションが拒否されると、オブジェクトの暗号化タイプは更新されますが、ユーザー定義タグはオブジェクトから削除され、エラーが発生します。

オブジェクトの暗号化を追加または変更するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、暗号化するオブジェクトが含まれるバケットの名前を選択します。
4. [オブジェクト] リストで、暗号化を追加または変更するオブジェクトの横のチェックボックスを選択します。

   オブジェクトの詳細ページが開き、オブジェクトのプロパティを表示するいくつかのセクションが表示されます。
5. [プロパティ] タブを選択します。
6. [デフォルトの暗号化] セクションまでスクロールして、[編集] を選択します。

   [デフォルトの暗号化を編集する] ページが開きます。
7. [暗号化タイプ] で、[AWS Key Management Service キーによる二層式サーバー側の暗号化 (DSSE-KMS)] を選択します。
8. [AWS KMS キー] で、次のいずれかを実行して KMS キーを選択します。

- 使用可能な KMS キーのリストから選択するには、[AWS KMS keys から選択する] を選択し、使用可能なキーのリストから自分の KMS キーを選択します。

  AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がこのリストに表示されます。カスタマーマネージドキーの詳細については、AWS Key Management Serviceデベロッパーガイドの「カスタマーキーと AWS キー」を参照してください。
- KMS キー ARN を入力するには、[AWS KMS key ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。
- AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

  AWS KMS key の作成の詳細については、AWS Key Management Service デベロッパーガイドのキーの作成を参照してください。

> **Important**
>
> バケットと同じ AWS リージョン で使用可能な KMS キーのみを使用できます。Amazon S3 コンソールには、バケットと同じリージョンで最初の 100 個の KMS キーしか表示されません。リストに存在しない KMS キーを使用するには、KMS キー ARN を入力する必要があります。別のアカウントが所有している KMS キーを使用する場合は、まず、そのキーを使用するアクセス許可が必要であり、次に、KMS キー ARN を入力する必要があります。
> Amazon S3 では、対称暗号化 KMS キーのみがサポートされ、非対称暗号化 KMS キーはサポートされません。詳細については、『AWS Key Management Service デベロッパーガイド』の「非対称 KMS キーの識別」を参照してください。

9. [バケットキー] で、[無効] を選択します。S3 バケットキーは DSSE-KMS ではサポートされていません。
10. [Save changes] (変更の保存) をクリックします。

> **Note**
>
> このアクションは、指定されたすべてのオブジェクトに暗号化を適用します。フォルダを暗号化する場合は、保存オペレーションが完了するのを待ってから、フォルダに新しいオブジェクトを追加します。

REST API の使用

オブジェクトを作成するとき (新しいオブジェクトをアップロードするか、既存のオブジェクトをコピーするとき) に、AWS KMS keys による二層式サーバー側の暗号化 (DSSE-KMS) の使用を指定してデータを暗号化できます。これを行うには、リクエストに `x-amz-server-side-encryption` ヘッダーを追加します。ヘッダーの値を、暗号化アルゴリズム `aws:kms:dsse` に設定します。Amazon S3 によりレスポンスヘッダー `x-amz-server-side-encryption` が返されるため、DSSE-KMS 暗号化を使用してオブジェクトが保存されたことを確認できます。

`aws:kms:dsse` の値で `x-amz-server-side-encryption` ヘッダーを指定する場合は、次のリクエストヘッダーも使用できます。

- `x-amz-server-side-encryption-aws-kms-key-id: SSEKMSKeyId`
- `x-amz-server-side-encryption-context: SSEKMSEncryptionContext`

トピック

- DSSE-KMS をサポートする Amazon S3 REST API オペレーション
- 暗号化コンテキスト (x-amz-server-side-encryption-context)
- AWS KMS キー ID (x-amz-server-side-encryption-aws-kms-key-id)

DSSE-KMS をサポートする Amazon S3 REST API オペレーション

次の REST API オペレーションでは `x-amz-server-side-encryption`、`x-amz-server-side-encryption-aws-kms-key-id`、および `x-amz-server-side-encryption-context` リクエストヘッダーを受け入れます。

- PutObject - PUT オペレーションを使用してデータをアップロードするとき、これらのリクエストヘッダーを指定できます。
- CopyObject — オブジェクトをコピーするときには、ソースオブジェクトとターゲットオブジェクトの両方があります。`CopyObject` オペレーションで DSSE-KMS ヘッダーを渡すと、ターゲットオブジェクトにのみ適用されます。既存のオブジェクトをコピーする際は、コピー元オブジェクトが暗号化されているかどうかに関係なく、明示的にサーバー側の暗号化を要求しない限り、コピー先オブジェクトは暗号化されません。
- POST オブジェクト — POST オペレーションを使用してオブジェクトをアップロードするときには、リクエストヘッダーの代わりに、フォームフィールドで同じ情報を指定します。

- CreateMultipartUpload - マルチパートアップロードを使用して大きいオブジェクトをアップロードするときには、CreateMultipartUpload リクエストでこれらのヘッダーを指定できます。

次の REST API オペレーションのレスポンスヘッダーは、オブジェクトがサーバー側の暗号化を使用して保存されるときに x-amz-server-side-encryption ヘッダーを返します。

- PutObject
- CopyObject
- POST オブジェクト
- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- GetObject
- HeadObject

> **Important**
>
> - AWS KMS で保護されたオブジェクトに対する GET および PUT リクエストはすべて、Secure Sockets Layer (SSL)、Transport Layer Security (TLS)、または署名バージョン 4 を使用していない場合、失敗します。
> - オブジェクトで DSSE-KMS を使用している場合、GET リクエストおよび HEAD リクエストの暗号化リクエストヘッダーを送信しないでください。送信すると、HTTP 400 (Bad Request) エラーが発生します。

暗号化コンテキスト (**x-amz-server-side-encryption-context**)

x-amz-server-side-encryption:aws:kms:dsse を指定した場合、Amazon S3 API は x-amz-server-side-encryption-context ヘッダーの暗号化コンテキストをサポートします。暗号化コンテキストは、データに関する追加のコンテキスト情報が含まれたキーバリューペアのセットです。

Amazon S3 は、オブジェクトの Amazon リソースネーム (ARN) を暗号化コンテキストペア (arn:aws:s3:::*object_ARN* など) として自動的に使用します。

オプションで、x-amz-server-side-encryption-context ヘッダーを使用して、追加の暗号化コンテキストペアを指定することもできます。しかし、暗号化コンテキストは暗号化されないため、機密情報を含めないでください。Amazon S3 は、この追加のキーペアをデフォルトの暗号化コンテキストとともに保存します。

Amazon S3 の暗号化コンテキストの詳細については、「暗号化コンテキスト」を参照してください。暗号化コンテキストの一般的な情報については、「AWS Key Management Service デベロッパーガイド」の「AWS Key Management Service Concepts - Encryption context」を参照してください。

AWS KMS キー ID (**x-amz-server-side-encryption-aws-kms-key-id**)

x-amz-server-side-encryption-aws-kms-key-id ヘッダーを使用して、データの保護に使用するカスタマーマネージドキーの ID を指定できます。x-amz-server-side-encryption:aws:kms:dsse ヘッダーを指定しても x-amz-server-side-encryption-aws-kms-key-id ヘッダーを指定しない場合、Amazon S3 は AWS マネージドキー (aws/s3) を使用してデータを保護します。カスタマーマネージドキーを使用する場合は、カスタマーマネージドキーの x-amz-server-side-encryption-aws-kms-key-id ヘッダーを指定する必要があります。

> **Important**
>
> Amazon S3 でサーバー側の暗号化に AWS KMS key を使用する場合は、対称暗号化 KMS キーを選択する必要があります。Amazon S3 は、対称暗号化 KMS キーのみをサポートします。このキーの詳細については、「AWS Key Management Service デベロッパーガイド」の「対称暗号化 KMS キー」を参照してください。

AWS CLI の使用

新しいオブジェクトをアップロードするか、既存のオブジェクトをコピーするときに、 データの暗号に DSSE-KMS を使用することを指定できます。これを行うには、リクエストに --server-side-encryption aws:kms:dsse パラメータを追加します。--ssekms-key-id *example-key-id* パラメータを使用して、作成したカスタマーマネージド AWS KMS キーを追加します。AWS KMS キー ID を指定せずに --server-side-encryption aws:kms:dsse を指定した場合、Amazon S3 は AWS マネージドキー (aws/s3) を使用します。

```
aws s3api put-object --bucket DOC-EXAMPLE-BUCKET --key example-object-key --server-side-encryption aws:kms:dsse --ssekms-key-id example-key-id --body filepath
```

暗号化されていないオブジェクトを元の場所にコピーすることで、DSSE-KMS を使用するように暗号化できます。

```
aws s3api copy-object --bucket DOC-EXAMPLE-BUCKET --key example-object-key --
body filepath --bucket DOC-EXAMPLE-BUCKET --key example-object-key --sse aws:kms:dsse
 --sse-kms-key-id example-key-id --body filepath
```

お客様が指定したキーによるサーバー側の暗号化 (SSE-C) の使用

サーバー側の暗号化は、保管時のデータ保護に関するものです。サーバー側の暗号化では、オブジェクトのメタデータではなく、オブジェクトデータのみが暗号化されます。お客様が用意したキーでのサーバー側の暗号化 (SSE-C) を使用すると、独自の暗号化キーを使用して暗号化されたデータを保存できます。リクエストの一部として用意された暗号化キーで、Amazon S3 は、ディスクに書き込む際のデータ暗号化と、オブジェクトにアクセスする際のデータ復号を管理します。したがって、データの暗号化と復号を実行するコードをお客様が管理する必要はありません。必要なことは、お客様が用意する暗号化キーを管理することだけです。

オブジェクトをアップロードする場合、Amazon S3 はお客様が用意した暗号化キーを使用してデータに AES-256 暗号化を適用します。その後、Amazon S3 はメモリから暗号化キーを削除します。オブジェクトを取得するときは、リクエストの中で同じ暗号化キーを指定する必要があります。Amazon S3 では、最初に指定された暗号化キーが一致することを確認した後、オブジェクトを復号してから、オブジェクトデータを返します。

SSE-C の使用に追加料金はかかりません。ただし、SSE-C を設定および使用するためのリクエストには、標準の Amazon S3 リクエスト料金が発生します。料金については、Amazon S3 の料金を参照してください。

> **Note**
>
> Amazon S3 では、お客様が用意した暗号化キーを保存しません。代わりに、以降のリクエストを検証するために、ランダムな SALT 値が付加された暗号化キーの Hash-based Message Authentication Code (HMAC) 値が保存されます。SALT 値が付加された HMAC 値を使用して、暗号化キーの値を求めたり、暗号化されたオブジェクトの内容を復号したりすることはできません。これは、暗号化キーを紛失した場合、オブジェクトが失われることを意味します。

S3 レプリケーションは、SSE-C で暗号化されたオブジェクトをサポートします。暗号化されたオブジェクトのレプリケーションの詳細については、「the section called “暗号化オブジェクトのレプリケート”」を参照してください。

SSE-C の詳細については、以下のトピックを参照してください。

トピック

- SSE-C の概要
- SSE-C の要件と制限
- 署名済み URL および SSE-C
- お客様が用意したキーによるサーバー側の暗号化 (SSE-C) の指定

SSE-C の概要

このセクションでは、SSE-C の概要を示します。SSE-C を使用する際は、以下の考慮事項に注意してください。

- HTTPS を使用する必要があります。

> **Important**
>
> SSE-C を使用する場合、Amazon S3 は HTTP 経由で行われたリクエストをすべて拒否します。セキュリティ上の考慮事項として、誤って HTTP で送信されたキーは漏洩したと見なすことをお勧めします。キーを破棄して、必要に応じて更新してください。

- レスポンスのエンティティタグ (ETag) はオブジェクトデータの MD5 ハッシュではありません。
- 使用した暗号化キーと暗号化したオブジェクトのマッピングは、お客様に管理していただきます。Amazon S3 では暗号化キーを保存しません。どのオブジェクトにどの暗号化キーを使用したかは、お客様が管理してください。
  - バケットのバージョニングが有効になっている場合、この機能を使用してアップロードする各オブジェクトバージョンに、独自の暗号化キーを使用できます。どのオブジェクトバージョンにどの暗号化キーを使用したかは、お客様が管理してください。
  - クライアント側の暗号化キーはお客様が管理するため、キーの更新など、クライアント側での追加の安全対策はお客様に管理していただきます。

> **Warning**
>
> 暗号化キーを紛失した場合、暗号化キーを使用せずにオブジェクトに対して GET リクエストを実行すると失敗し、オブジェクトは失われます。

SSE-C の要件と制限

特定の Amazon S3 バケット内のすべてのオブジェクトの SSE-C を要求するには、バケットポリシーを使用できます。

例えば、次のバケットポリシーは、SSE-C を要求する `x-amz-server-side-encryption-customer-algorithm` ヘッダーを含まないすべてのリクエストに対して、オブジェクトのアップロード (`s3:PutObject`) 許可を拒否します。

```
{
    "Version": "2012-10-17",
    "Id": "PutObjectPolicy",
    "Statement": [
        {
            "Sid": "RequireSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::example-s3-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "true"
                }
            }
        }
    ]
}
```

特定の Amazon S3 バケット内のすべてのオブジェクトのサーバー側の暗号化を制限する場合も、ポリシーを使用できます。例えば、SSE-C を要求する `x-amz-server-side-encryption-customer-algorithm` ヘッダーがリクエストに含まれている場合、次のバケットポリシーはすべてのユーザーに対し、オブジェクト (`s3:PutObject`) をアップロードするアクセス許可を拒否します。

```
{
    "Version": "2012-10-17",
    "Id": "PutObjectPolicy",
    "Statement": [
        {
            "Sid": "RestrictSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::example-s3-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "false"
                }
            }
        }
    ]
}
```

> **Important**
>
> バケットポリシーを使用して s3:PutObject に対する SSE-C を要求する場合は、すべてのマルチパートアップロードリクエスト (CreateMultiPartUpload、UploadPart、CompleteMultipartUpload) に x-amz-server-side-encryption-customer-algorithm ヘッダーを含める必要があります。

署名済み URL および SSE-C

新しいオブジェクトのアップロード、既存のオブジェクトやオブジェクトメタデータの取得などのオペレーションに使用できる署名付き URL を生成できます。署名付き URL では、次のように SSE-C がサポートされます。

- 署名付き URL を作成するときに、署名の計算で x-amz-server-side-encryption-customer-algorithm ヘッダーを使用してアルゴリズムを指定する必要があります。
- 署名付き URL を使用して新しいオブジェクトをアップロードするとき、既存のオブジェクトを取得するとき、またはオブジェクトメタデータのみを取得するときに、クライアントアプリケーションのリクエストですべての暗号化ヘッダーを指定する必要があります。

> **Note**
>
> SSE-C 以外のオブジェクトでは、署名付き URL を生成し、その URL をブラウザに直接貼り付けることで、データにアクセスできます。
> ただし、これは SSE-C オブジェクトには行えません。署名付き URL に加えて SSE-C オブジェクトに固有の HTTP ヘッダーも含める必要があります。したがって、SSE-C オブジェクトの署名付き URL はプログラムでのみ使用できます。

署名付き URL の詳細については、「the section called “署名付き URL の使用”」を参照してください。

お客様が用意したキーによるサーバー側の暗号化 (SSE-C) の指定

REST API を使用したオブジェクトの作成時に、お客様が用意したキー (SSE-C) を使用してサーバー側の暗号化を指定できます。SSE-C を使用する場合は、次のリクエストヘッダーを使用して暗号化キー情報を指定する必要があります。

名前	説明
`x-amz-server-side-encryption-customer-algorithm`	暗号化アルゴリズムを指定するには、このヘッダーを使用します。ヘッダーの値は `AES256` である必要があります。
`x-amz-server-side-encryption-customer-key`	Amazon S3 でデータを暗号化または復号するために使用する base64 でエンコードされた 256 ビットの暗号化キーを指定するには、このヘッダーを使用します。
`x-amz-server-side-encryption-customer-key-MD5`	RFC 1321 に従って、暗号化キーの base64 エンコードされた 128 ビット MD5 ダイジェストを指定するには、このヘッダーを使用します。Amazon S3 では、このヘッダーを使用してメッセージの整合性を調べて、送信された暗号化キーにエラーがないことを確認します。

AWS SDK ラッパーライブラリを使用して、これらのヘッダーをリクエストに追加できます。必要に応じて、アプリケーションから直接 Amazon S3 REST API を呼び出すことができます。

> **Note**
>
> Amazon S3 コンソールを使用してオブジェクトをアップロードしたり SSE-C をリクエストしたりすることはできません。また、SSE-C を使用して保存されている既存のオブジェクトを更新すること (ストレージクラスの変更やメタデータの追加など) もできません。

REST API の使用

SSE-C をサポートする Amazon S3 REST API

次の Amazon S3 API は、お客様が用意した暗号化キー (SSE-C) を使用したサーバー側の暗号化をサポートします。

- GET オペレーション — GET API (GET Object を参照) を使用してオブジェクトを取得するときに、このリクエストヘッダーを指定できます。
- HEAD オペレーション — HEAD API (HEAD Object を参照) を使用してオブジェクトメタデータを取得するには、これらのリクエストヘッダーを指定できます。
- PUT オペレーション — PUT API (PUT Object を参照) を使用してデータをアップロードするときに、これらのリクエストヘッダーを指定できます。
- マルチパートアップロード — マルチパートアップロード API を使用して大きいオブジェクトをアップロードするときに、これらのヘッダーを指定できます。これらのヘッダーは、開始リクエスト (Initiate Multipart Upload を参照) と、後続の各パートのアップロードリクエスト (Upload Part または Upload Part - Copy を参照) で指定します。各パートのアップロードリクエストでは、暗号化情報がマルチパートアップロードの開始リクエストで指定した情報と同じである必要があります。
- POST オペレーション — POST オペレーションを使用してオブジェクトをアップロードする場合は (POST Object を参照)、リクエストヘッダーの代わりに、フォームフィールドで同じ情報を指定します。
- Copy オペレーション — オブジェクトをコピーする場合 (PUT Object - Copy を参照)、ソースオブジェクトとターゲットオブジェクトがあります。
  - AWS 管理のキーによるサーバー側の暗号化を使用してターゲットオブジェクトを暗号化する場合は、`x-amz-server-side-encryption` リクエストヘッダーを指定する必要があります。
  - SSE-C を使用してターゲットオブジェクトを暗号化する場合は、前の表で説明した 3 つのヘッダーを使用して暗号化情報を指定する必要があります。

- ソースオブジェクトが SSE-C を使用して暗号化されている場合、Amazon S3 でオブジェクトを復号してコピーできるように、次のヘッダーを使用して暗号化キーの情報を指定する必要があります。

名前	説明
`x-amz-copy-source-server-side-encryption-customer-algorithm`	Amazon S3 でソースオブジェクトを復号するために使用するアルゴリズムを指定するには、このヘッダーを含めます。この値は、`AES256` である必要があります。
`x-amz-copy-source-server-side-encryption-customer-key`	Amazon S3 でソースオブジェクトを復号するために使用する base64 でエンコードされた暗号化キーを指定するには、このヘッダーを含めます。この暗号化キーは、Amazon S3 でソースオブジェクトを作成したときに指定したキーであることが必要です。それ以外の場合、Amazon S3 でオブジェクトを復号できません。
`x-amz-copy-source-server-side-encryption-customer-key-MD5`	RFC 1321 に従って、暗号化キーの base64 エンコードされた 128 ビット MD5 ダイジェストを指定するには、このヘッダーを含めます。

AWS SDK を使用して PUT、GET、Head、および Copy オペレーションの SSE-C を指定する

次の例では、お客様が用意したキーによるサーバー側の暗号化 (SSE-C) をオブジェクト用にリクエストする方法を示します。この例では、次の操作を実行します。各オペレーションでは、SSE-C 関連ヘッダーをリクエストで指定する方法を示します。

- Put object — オブジェクトをアップロードし、顧客が用意した暗号キーによるサーバー側の暗号化をリクエストします。
- Get object — 前のステップでアップロードしたオブジェクトをダウンロードします。リクエストでは、オブジェクトのアップロード時に指定したのと同じ暗号化情報を提供します。Amazon S3 は、オブジェクトを復号して返すために、この情報を必要とします。

- Get object metadata — オブジェクトのメタデータを取得します。オブジェクトの作成時に使用したのと同じ暗号化情報を指定します。
- Copy object — 以前にアップロードしたオブジェクトのコピーを作成します。ソースオブジェクトは SSE-C を使用して保存されるため、コピーリクエストで暗号化情報を指定する必要があります。デフォルトでは、明示的にリクエストした場合に限り、Amazon S3 はオブジェクトのコピーを暗号化します。この例では、オブジェクトの暗号化されたコピーを保存するように Amazon S3 に指示します。

Java

> **Note**
>
> この例では、1 つのオペレーションでオブジェクトをアップロードする方法を示します。マルチパートアップロード API を使用して大きなオブジェクトをアップロードする場合は、この例に示したのと同じ方法で暗号化情報を指定します。AWS SDK for Java を使用するマルチパートアップロードの例については、マルチパートアップロードを使用したオブジェクトのアップロード を参照してください。

必要な暗号化情報を追加するには、リクエストに `SSECustomerKey` を含めます。`SSECustomerKey` クラスの詳細については、REST API」セクションを参照してください。

SSE-C の詳細については、お客様が指定したキーによるサーバー側の暗号化 (SSE-C) の使用 を参照してください。作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

Example

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import javax.crypto.KeyGenerator;
import java.io.BufferedReader;
import java.io.File;
```

```
import java.io.IOException;
import java.io.InputStreamReader;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

public class ServerSideEncryptionUsingClientSideEncryptionKey {
    private static SSECustomerKey SSE_KEY;
    private static AmazonS3 S3_CLIENT;
    private static KeyGenerator KEY_GENERATOR;

    public static void main(String[] args) throws IOException,
 NoSuchAlgorithmException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String uploadFileName = "*** File path ***";
        String targetKeyName = "*** Target key name ***";

        // Create an encryption key.
        KEY_GENERATOR = KeyGenerator.getInstance("AES");
        KEY_GENERATOR.init(256, new SecureRandom());
        SSE_KEY = new SSECustomerKey(KEY_GENERATOR.generateKey());

        try {
            S3_CLIENT = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Upload an object.
            uploadObject(bucketName, keyName, new File(uploadFileName));

            // Download the object.
            downloadObject(bucketName, keyName);

            // Verify that the object is properly encrypted by attempting to
 retrieve it
            // using the encryption key.
            retrieveObjectMetadata(bucketName, keyName);

            // Copy the object into a new object that also uses SSE-C.
            copyObject(bucketName, keyName, targetKeyName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
```

```
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    private static void uploadObject(String bucketName, String keyName, File file) {
        PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName,
file).withSSECustomerKey(SSE_KEY);
        S3_CLIENT.putObject(putRequest);
        System.out.println("Object uploaded");
    }

    private static void downloadObject(String bucketName, String keyName) throws
IOException {
        GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName,
keyName).withSSECustomerKey(SSE_KEY);
        S3Object object = S3_CLIENT.getObject(getObjectRequest);

        System.out.println("Object content: ");
        displayTextInputStream(object.getObjectContent());
    }

    private static void retrieveObjectMetadata(String bucketName, String keyName) {
        GetObjectMetadataRequest getMetadataRequest = new
GetObjectMetadataRequest(bucketName, keyName)
                .withSSECustomerKey(SSE_KEY);
        ObjectMetadata objectMetadata =
S3_CLIENT.getObjectMetadata(getMetadataRequest);
        System.out.println("Metadata retrieved. Object size: " +
objectMetadata.getContentLength());
    }

    private static void copyObject(String bucketName, String keyName, String
targetKeyName)
            throws NoSuchAlgorithmException {
        // Create a new encryption key for target so that the target is saved using
        // SSE-C.
        SSECustomerKey newSSEKey = new SSECustomerKey(KEY_GENERATOR.generateKey());
```

```
        CopyObjectRequest copyRequest = new CopyObjectRequest(bucketName, keyName,
 bucketName, targetKeyName)
                .withSourceSSECustomerKey(SSE_KEY)
                .withDestinationSSECustomerKey(newSSEKey);

        S3_CLIENT.copyObject(copyRequest);
        System.out.println("Object copied");
    }

    private static void displayTextInputStream(S3ObjectInputStream input) throws
 IOException {
        // Read one line at a time from the input stream and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

.NET

> **Note**
>
> マルチパートアップロード API を使用した大きなオブジェクトのアップロードの例については、マルチパートアップロードを使用したオブジェクトのアップロード および AWS SDK の使用 (低レベル API) を参照してください。

SSE-C の詳細については、お客様が指定したキーによるサーバー側の暗号化 (SSE-C) の使用 を参照してください。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.IO;
```

```
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSEClientEncryptionKeyObjectOperationsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for new object created ***";
        private const string copyTargetKeyName = "*** key name for object copy ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            ObjectOpsUsingClientEncryptionKeyAsync().Wait();
        }
        private static async Task ObjectOpsUsingClientEncryptionKeyAsync()
        {
            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // 1. Upload the object.
                PutObjectRequest putObjectRequest = await
 UploadObjectAsync(base64Key);
                // 2. Download the object and verify that its contents matches what
 you uploaded.
                await DownloadObjectAsync(base64Key, putObjectRequest);
                // 3. Get object metadata and verify that the object uses AES-256
 encryption.
                await GetObjectMetadataAsync(base64Key);
                // 4. Copy both the source and target objects using server-side
 encryption with
                //    a customer-provided encryption key.
                await CopyObjectAsync(aesEncryption, base64Key);
            }
```

```
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        private static async Task<PutObjectRequest> UploadObjectAsync(string
base64Key)
        {
            PutObjectRequest putObjectRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };
            PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }
        private static async Task DownloadObjectAsync(string base64Key,
PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                // Provide encryption information for the object stored in Amazon
S3.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
```

```
            using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (String.Compare(putObjectRequest.ContentBody, content) == 0)
                    Console.WriteLine("Object content is same as we uploaded");
                else
                    Console.WriteLine("Error...Object content is not same.");

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                    Console.WriteLine("Object encryption method is AES256, same as
we set");
                else
                    Console.WriteLine("Error...Object encryption method is not the
same as AES256 we set");

                // Assert.AreEqual(putObjectRequest.ContentBody, content);
                // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getResponse.ServerSideEncryptionCustomerMethod);
            }
        }
        private static async Task GetObjectMetadataAsync(string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used is:
{0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
            // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }
```

```
        private static async Task CopyObjectAsync(Aes aesEncryption, string
 base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
                DestinationKey = copyTargetKeyName,
                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,
                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key
            };
            await client.CopyObjectAsync(copyRequest);
        }
    }
}
```

AWS SDK を使用してマルチパートアップロードの SSE-C を指定する

前のセクションの例では、PUT、GET、Head、および COPY オペレーションで、お客様が用意したキーによるサーバー側の暗号化 (SSE-C) をリクエストする方法を示しています。このセクションでは、SSE-C をサポートするその他の Amazon S3 API について説明します。

Java

大きなオブジェクトをアップロードするために、マルチパートアップロード API を使用できます (マルチパートアップロードを使用したオブジェクトのアップロードとコピー を参照)。高レベル API または低レベル API を使用して、大きなオブジェクトをアップロードできます。これらの API は、リクエストでの暗号化関連のヘッダーをサポートします。

- 高レベルの TransferManager API を使用する場合は、PutObjectRequest で暗号化専用のヘッダーを指定します (「マルチパートアップロードを使用したオブジェクトのアップロード」を参照)。
- 低レベル API を使用する場合は、暗号化関連情報を InitiateMultipartUploadRequest で指定し、続けて各 UploadPartRequest で同じ暗号化情報を指定します。CompleteMultipartUploadRequest で暗号化専用のヘッダーを指定する必要はありません。例については、AWS SDK の使用 (低レベル API) を参照してください。

次の例では、TransferManager を使用してオブジェクトを作成し、SSE-C 関連の情報を提供する方法を示します。この例では、次のような処理を実行します。

- TransferManager.upload() メソッドを使用してオブジェクトを作成します。PutObjectRequest インスタンスで、リクエストする暗号キーの情報を指定します。Amazon S3 は、お客様が用意したキーを使用してオブジェクトを暗号化します。
- TransferManager.copy() メソッドを呼び出してオブジェクトのコピーを作成します。この例では、新しい SSECustomerKey を使用してオブジェクトのコピーを暗号化するように Amazon S3 に指示します。ソースオブジェクトは SSE-C で暗号化されているため、CopyObjectRequest はソースオブジェクトの暗号化キーを提供し、Amazon S3 で復号してからコピーできるようにします。

Example

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.SSECustomerKey;
import com.amazonaws.services.s3.transfer.Copy;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import javax.crypto.KeyGenerator;
import java.io.File;
```

```
import java.security.SecureRandom;

public class ServerSideEncryptionCopyObjectUsingHLwithSSEC {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String fileToUpload = "*** File path ***";
        String keyName = "*** New object key name ***";
        String targetKeyName = "*** Key name for object copy ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
            TransferManager tm = TransferManagerBuilder.standard()
                    .withS3Client(s3Client)
                    .build();

            // Create an object from a file.
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName,
 keyName, new File(fileToUpload));

            // Create an encryption key.
            KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
            keyGenerator.init(256, new SecureRandom());
            SSECustomerKey sseCustomerEncryptionKey = new
 SSECustomerKey(keyGenerator.generateKey());

            // Upload the object. TransferManager uploads asynchronously, so this
 call
            // returns immediately.
            putObjectRequest.setSSECustomerKey(sseCustomerEncryptionKey);
            Upload upload = tm.upload(putObjectRequest);

            // Optionally, wait for the upload to finish before continuing.
            upload.waitForCompletion();
            System.out.println("Object created.");

            // Copy the object and store the copy using SSE-C with a new key.
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucketName,
 keyName, bucketName, targetKeyName);
```

```
            SSECustomerKey sseTargetObjectEncryptionKey = new
SSECustomerKey(keyGenerator.generateKey());
            copyObjectRequest.setSourceSSECustomerKey(sseCustomerEncryptionKey);

copyObjectRequest.setDestinationSSECustomerKey(sseTargetObjectEncryptionKey);

            // Copy the object. TransferManager copies asynchronously, so this call
returns
            // immediately.
            Copy copy = tm.copy(copyObjectRequest);

            // Optionally, wait for the upload to finish before continuing.
            copy.waitForCompletion();
            System.out.println("Copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

大きなオブジェクトをアップロードするために、マルチパートアップロード API を使用できます (「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照)。AWSSDK for .NET には、大きなオブジェクトをアップロードするため、高レベルおよび低レベルの両方の API が用意されています。これらの API は、リクエストでの暗号化関連のヘッダーをサポートします。

- 高レベルの Transfer-Utility API を使用するときには、次に示すように TransferUtilityUploadRequest で暗号化固有のヘッダーを提供します。コード例については、マルチパートアップロードを使用したオブジェクトのアップロード を参照してください。

```
TransferUtilityUploadRequest request = new TransferUtilityUploadRequest()
{
```

```
    FilePath = filePath,
    BucketName = existingBucketName,
    Key = keyName,
    // Provide encryption information.
    ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
    ServerSideEncryptionCustomerProvidedKey = base64Key,
};
```

- 低レベル API を使用する場合は、マルチパートアップロードの開始リクエストで暗号化関連情報を提供し、以降のパートのアップロードリクエストでは同じ暗号化情報を提供します。マルチパートアップロードの完了リクエストでは、暗号化固有のヘッダーを提供する必要はありません。例については、AWS SDK の使用 (低レベル API) を参照してください。

  既存の大きなオブジェクトをコピーする低レベルのマルチパートアップロードの例を次に示します。この例では、コピーされるオブジェクトは SSE-C を使用して Amazon S3 に保存されており、ターゲットオブジェクトも SSE-C を使用して保存します。例えば、以下を実行します。

  - 暗号化キーと関連情報を提供することによって、マルチパートアップロードのリクエストを開始します。
  - CopyPartRequest でソースオブジェクトとターゲットオブジェクトの暗号化キーを提供します。
  - オブジェクトメタデータを取得することによって、コピーされるソースオブジェクトのサイズを取得します。
  - 5 MB のパート単位でオブジェクトをアップロードします。

  Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSECLowLevelMPUcopyObjectTest
```

```
   {
        private const string existingBucketName = "*** bucket name ***";
        private const string sourceKeyName      = "*** source object key name
***";
        private const string targetKeyName      = "*** key name for the target
object ***";
        private const string filePath           = @"*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CopyObjClientEncryptionKeyAsync().Wait();
        }

        private static async Task CopyObjClientEncryptionKeyAsync()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);

            await CreateSampleObjUsingClientEncryptionKeyAsync(base64Key,
s3Client);

            await CopyObjectAsync(s3Client, base64Key);
        }
        private static async Task CopyObjectAsync(IAmazonS3 s3Client, string
base64Key)
        {
            List<CopyPartResponse> uploadResponses = new List<CopyPartResponse>();

            // 1. Initialize.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = targetKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };
```

```
            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // 2. Upload Parts.
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB
            long firstByte = 0;
            long lastByte = partSize;

            try
            {
                // First find source object size. Because object is stored
encrypted with
                // customer provided key you need to provide encryption
information in your request.
                GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest()
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                    ServerSideEncryptionCustomerProvidedKey = base64Key // " *
**source object encryption key ***"
                };

                GetObjectMetadataResponse getObjectMetadataResponse = await
s3Client.GetObjectMetadataAsync(getObjectMetadataRequest);

                long filePosition = 0;
                for (int i = 1; filePosition <
getObjectMetadataResponse.ContentLength; i++)
                {
                    CopyPartRequest copyPartRequest = new CopyPartRequest
                    {
                        UploadId = initResponse.UploadId,
                        // Source.
                        SourceBucket = existingBucketName,
                        SourceKey = sourceKeyName,
                        // Source object is stored using SSE-C. Provide encryption
information.
                        CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
```

```
                        CopySourceServerSideEncryptionCustomerProvidedKey =
base64Key, //"***source object encryption key ***",
                        FirstByte = firstByte,
                        // If the last part is smaller then our normal part size
then use the remaining size.
                        LastByte = lastByte >
getObjectMetadataResponse.ContentLength ?
                            getObjectMetadataResponse.ContentLength - 1 :
lastByte,

                        // Target.
                        DestinationBucket = existingBucketName,
                        DestinationKey = targetKeyName,
                        PartNumber = i,
                        // Encryption information for the target object.
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key
                    };
                    uploadResponses.Add(await
s3Client.CopyPartAsync(copyPartRequest));
                    filePosition += partSize;
                    firstByte += partSize;
                    lastByte += partSize;
                }

                // Step 3: complete.
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = targetKeyName,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine("Exception occurred: {0}", exception.Message);
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
```

```
                {
                    BucketName = existingBucketName,
                    Key = targetKeyName,
                    UploadId = initResponse.UploadId
                };
                s3Client.AbortMultipartUpload(abortMPURequest);
            }
        }
        private static async Task
CreateSampleObjUsingClientEncryptionKeyAsync(string base64Key, IAmazonS3
s3Client)
        {
            // List to store upload part responses.
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

            // 1. Initialize.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            InitiateMultipartUploadResponse initResponse =
               await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // 2. Upload Parts.
            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
```

```
                        PartNumber = i,
                        PartSize = partSize,
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
s3Client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                // Step 3: complete.
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                    //PartETags = new List<PartETag>(uploadResponses)

                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);

            }
            catch (Exception exception)
            {
                Console.WriteLine("Exception occurred: {0}", exception.Message);
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId
                };
                await s3Client.AbortMultipartUploadAsync(abortMPURequest);
            }
```

```
        }
    }
}
```

クライアント側の暗号化を使用したデータの保護

クライアント側の暗号化は、データをローカルで暗号化することで、転送時および保管時のセキュリティを確保するためのものです。Amazon S3 に送信する前にオブジェクトを暗号化するには、Amazon S3 暗号化クライアントを使用します。オブジェクトがこの方法で暗号化されている場合、オブジェクトは AWS を含むサードパーティーに公開されません。Amazon S3 は、すでに暗号化されているオブジェクトを受け取るだけで、オブジェクトの暗号化または復号には関与しません。Amazon S3 暗号化クライアントとサーバー側の暗号化の両方を使用してデータを暗号化できます。暗号化されたオブジェクトを Amazon S3 に送信すると、Amazon S3 はオブジェクトを暗号化されているとは認識せず、一般的なオブジェクトのみを検出します。

Amazon S3 暗号化クライアントは、お客様と Amazon S3 の間の仲介役として機能します。Amazon S3 暗号化クライアントをインスタンス化すると、オブジェクトは、Amazon S3 `PutObject` および `GetObject` リクエストの一部として自動的に暗号化および復号化されます。オブジェクトはすべて、一意のデータキーで暗号化されます。Amazon S3 暗号化クライアントは、ラップキーとして KMS キーを指定した場合でも、バケットキーを使用したり、バケットキーを操作したりしません。

Amazon S3 暗号化クライアント開発者ガイドは、Amazon S3 暗号化クライアントのバージョン 3.0 以降に焦点を当てています。詳細については、Amazon S3 暗号化クライアントデベロッパーガイドの「What is the Amazon S3 Encryption Client?」(Amazon S3 暗号化クライアントとは) を参照してください。

Amazon S3 暗号化クライアントの以前のバージョンの詳細については、プログラミング言語に応じた AWS SDK デベロッパーガイドを参照してください。

- AWS SDK for Java
- AWS SDK for .NET
- AWS SDK for Go
- AWS SDK for PHP
- AWS SDK for Ruby
- AWS SDK for C++

インターネットトラフィックのプライバシー

このトピックでは、Amazon S3 でサービスから他の口ケーションまでの接続を保護する方法について説明します。

サービスとオンプレミスのクライアントおよびアプリケーションとの間のトラフィック

次の接続は、プライベートネットワークと AWS PrivateLink との間の接続を提供する AWS と組み合わせられます。

- AWS Site−to−Site VPN 接続。詳細については、AWS Site-to-Site VPN とは何ですか? を参照してください。
- AWS Direct Connect 接続。詳細については、AWS Direct Connect とは を参照してください。

ネットワークを経由した Amazon S3 へのアクセスは、AWS が発行する API を介して行われます。クライアントは Transport Layer Security (TLS) 1.2 をサポートしている必要があります。TLS 1.3 をお勧めします。クライアントは、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) などの Perfect Forward Secrecy (PFS) を備えた暗号スイートもサポートする必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。また、リクエストには、IAM プリンシパルに関連付けられたアクセスキー ID およびシークレットアクセスキーによる署名が必要です。または、リクエストへの署名のために一時的にセキュリティ認証情報を生成する AWS Security Token Service (STS) を使用することもできます。

同じリージョン内の AWS リソース間のトラフィック

Amazon S3 の 仮想プライベートクラウド (VPC) エンドポイントは、Amazon S3 への接続のみを許可する VPC 内の論理エンティティです。VPC はリクエストを Amazon S3 にルーティングし、レスポンスを VPC にルーティングします。詳細については、VPC ユーザーガイドの VPC エンドポイント を参照してください。VPC エンドポイントから S3 バケットへのアクセスをコントロールするために使用できるバケットポリシーの例については、バケットポリシーを使用した VPC エンドポイントからのアクセスコントロール を参照してください。

AWS PrivateLink for Amazon S3

AWS PrivateLink for Amazon S3 では、Virtual Private Cloud (VPC) でインターフェイス VPC エンドポイント (インターフェイスエンドポイント) をプロビジョニングできます。これらのエンドポイン

トは、VPN および AWS Direct Connect 経由でオンプレミスにあるアプリケーション、または VPC ピアリング経由で別の AWS リージョン にあるアプリケーションから直接アクセスできます。

インターフェイスエンドポイントは、VPC 内のサブネットからプライベート IP アドレスが割り当てられた 1 つ以上の Elastic Network Interface (ENI) で表されます。インターフェイスエンドポイントを介した Amazon S3 へのリクエストは、Amazon ネットワークに残ります。AWS Direct Connect または AWS Virtual Private Network (AWS VPN) を介して、オンプレミスのアプリケーションから VPC 内のインターフェイスエンドポイントにアクセスすることもできます。VPC をオンプレミスネットワークに接続する方法の詳細については、AWS Direct Connect ユーザーガイドおよび AWS Site-to-Site VPN ユーザーガイドを参照してください。

インターフェイスエンドポイントの一般的な情報については、AWS PrivateLink ガイドのインターフェイス VPC エンドポイント (AWS PrivateLink) を参照してください。

トピック

- Amazon S3 の VPC エンドポイントのタイプ
- AWS PrivateLink for Amazon S3 の制約と制限
- VPC エンドポイントの作成
- Amazon S3 インターフェイスエンドポイントへのアクセス
- プライベート DNS
- S3 インターフェイスエンドポイントからバケット、アクセスポイント、および Amazon S3 コントロール API オペレーションにアクセスする
- オンプレミスの DNS 設定の更新
- Amazon S3 用の VPC エンドポイントポリシーの作成

Amazon S3 の VPC エンドポイントのタイプ

Amazon S3 へのアクセスには、ゲートウェイエンドポイントとインターフェイスエンドポイント (AWS PrivateLink を使用) の 2 つのタイプの VPC エンドポイントを使用できます。ゲートウェイエンドポイントは、AWS ネットワーク経由で VPC から Amazon S3 にアクセスするために、ルートテーブルで指定するゲートウェイです。インターフェイスエンドポイントは、プライベート IP アドレスを使用して、VPC 内、オンプレミス、または VPC ピアリングや AWS リージョン を使用する別の AWS Transit Gateway にある VPC から Amazon S3 にリクエストをルーティングすることにより、ゲートウェイエンドポイントの機能を拡張します。詳細については、「VPC ピア機能とは」および「Transit Gateway vs VPC peering」(Transit Gateway と VPC ピアリング) を参照してください。

インターフェイスエンドポイントは、ゲートウェイエンドポイントと互換性があります。VPC 内に既存のゲートウェイエンドポイントがある場合は、同じ VPC で両方のタイプのエンドポイントを使用できます。

Amazon S3 のゲートウェイエンドポイント	Amazon S3 のインターフェイスエンドポイント
いずれの場合も、ネットワークトラフィックは AWS ネットワーク上に残ります。	
Amazon S3 パブリック IP アドレスを使用する	VPC のプライベート IP アドレスを使用して Amazon S3 にアクセスする
同じ Amazon S3 DNS 名を使用する	エンドポイント固有の Amazon S3 DNS 名を要求する
オンプレミスからのアクセスを許可しない	オンプレミスからのアクセスを許可する
別の AWS リージョンからのアクセスを許可しない	VPC ピアリングまたは AWS Transit Gateway を使用する別の AWS リージョンにある VPC からのアクセスを許可する
課金されない	請求される

ゲートウェイエンドポイントの詳細については、AWS PrivateLink ガイドのゲートウェイ VPC エンドポイントを参照してください。

AWS PrivateLink for Amazon S3 の制約と制限

VPC 制限は、AWS PrivateLink for Amazon S3 に適用されます。詳細については、AWS PrivateLink ガイドの「インターフェイスエンドポイントの考慮事項」と「AWS PrivateLink クォータ」を参照してください。また、以下の制約も適用されます。

AWS PrivateLink for Amazon S3 では、以下はサポートされていません。

- 連邦情報処理規格 (FIPS) エンドポイント
- ウェブサイトエンドポイント
- レガシーグローバルエンドポイント
- S3 ダッシュリージョンのエンドポイント

- Amazon S3 デュアルスタックエンドポイント
- 異なる AWS リージョン のバケット間での CopyObject または UploadPartCopy の使用
- Transport Layer Security (TLS) 1.1

VPC エンドポイントの作成

VPC インターフェイスエンドポイントを作成するには、AWS PrivateLink ガイドの「VPC エンドポイントの作成」を参照してください。

Amazon S3 インターフェイスエンドポイントへのアクセス

インターフェイスエンドポイントを作成すると、Amazon S3 はエンドポイント固有の 2 つのタイプの S3 DNS 名 (Regional および zonal) を生成します。

- Regional DNS 名には、一意の VPC エンドポイント ID、サービス識別子、AWS リージョン、およびその名前の `vpce.amazonaws.com` が含まれます。例えば、VPC エンドポイント ID *`vpce-1a2b3c4d`* の場合、生成される DNS 名は *`vpce-1a2b3c4d-5e6f`*`.s3.us-east-1.vpce.amazonaws.com` に似ている場合があります。
- Zonal DNS 名には、アベイラビリティーゾーンが含まれます (*`vpce-1a2b3c4d-5e6f`*`-`**`us-east-1a`**`.s3.us-east-1.vpce.amazonaws.com` など)。このオプションは、アーキテクチャがアベイラビリティーゾーンを分離する場合に使用できます。例えば、障害を隔離し、リージョン間のデータ転送コストを削減するために使用できます。

エンドポイント固有の S3 DNS 名は、S3 パブリック DNS ドメインから解決できます。

プライベート DNS

VPC インターフェイスエンドポイント用のプライベート DNS オプションを使用すると、VPC エンドポイント経由の S3 トラフィックのルーティングが簡単になり、アプリケーションで使用できる最も低コストのネットワークパスを活用できます。プライベート DNS オプションを使用すると、インターフェイスエンドポイントのエンドポイント固有の DNS 名を使用するように S3 クライアントを更新したり、DNS インフラストラクチャを管理したりすることなく、Regional S3 トラフィックをルーティングできます。プライベート DNS 名を有効にすると、Regional S3 DNS クエリは次のエンドポイントの AWS PrivateLink のプライベート IP アドレスに解決されます。

- Regional バケットエンドポイント (例: `s3.us-east-1.amazonaws.com`)
- 制御エンドポイント (例: `s3-control.us-east-1.amazonaws.com`)

- アクセスポイントエンドポイント (例: `s3-accesspoint.us-east-1.amazonaws.com`)

VPC にゲートウェイエンドポイントがある場合は、VPC 内リクエストを既存の S3 ゲートウェイエンドポイントに、オンプレミスリクエストをインターフェイスエンドポイントに自動的にルーティングできます。このアプローチでは、VPC 内のトラフィックに課金されないゲートウェイエンドポイントを使用することで、ネットワークコストを最適化できます。オンプレミスアプリケーションは、インバウンドリゾルバーエンドポイントを利用して AWS PrivateLink を使用できます。Amazon は、「Route 53 Resolver」と呼ばれる VPC 用の DNS サーバーを提供しています。インバウンドリゾルバーエンドポイントは、DNS クエリをオンプレミスネットワークから Route 53 Resolver に転送します。

> **Important**
>
> [インバウンドエンドポイントでのみプライベート DNS を有効にする] を使用するときに最も低コストのネットワークパスを利用するには、ゲートウェイエンドポイントが VPC に存在している必要があります。ゲートウェイエンドポイントが存在すると、[インバウンドエンドポイントでのみプライベート DNS を有効にする] オプションが選択されている場合でも、VPC 内のトラフィックは常に AWS プライベートネットワーク経由でルーティングされます。[インバウンドエンドポイントでのみプライベート DNS を有効にする] オプションが選択されている間は、このゲートウェイエンドポイントを維持する必要があります。ゲートウェイエンドポイントを削除する場合は、まず [インバウンドエンドポイントでのみプライベート DNS を有効にする] をオフにする必要があります。
> 既存のインターフェイスエンドポイントを [インバウンドエンドポイントでのみプライベート DNS を有効にする] に更新する場合は、まず VPC に S3 ゲートウェイエンドポイントがあることを確認します。ゲートウェイエンドポイントとプライベート DNS 名の管理の詳細については、AWS PrivateLink ガイドの「ゲートウェイ VPC エンドポイント」と「VPC エンドポイントサービスの DNS 名を管理する」をそれぞれ参照してください。

[インバウンドエンドポイントでのみプライベート DNS を有効にする] オプションは、ゲートウェイエンドポイントをサポートするサービスでのみ使用できます。

[インバウンドエンドポイントでのみプライベート DNS を有効にする] を使用する VPC エンドポイントの作成の詳細については、AWS PrivateLink ガイドのの「VPC エンドポイントの作成」を参照してください。

Amazon VPC コンソールの使用

コンソールには、[DNS 名を有効にする] と [インバウンドエンドポイントに対してのみプライベート DNS を有効にする] の 2 つのオプションがあります。[DNS 名を有効にする] は、AWS PrivateLink でサポートされているオプションです。[DNS 名を有効にする] オプションを使用すると、デフォルトのパブリックエンドポイント DNS 名にリクエストを送信しながら、Amazon S3 への Amazon のプライベート接続を使用できます。このオプションを有効にすると、お客様はアプリケーションで使用できる最も低コストのネットワークパスを利用できます。

Amazon S3 の既存または新しい VPC インターフェイスエンドポイントでプライベート DNS 名を有効にすると、デフォルトで [インバウンドエンドポイントに対してのみプライベート DNS を有効にする] オプションが選択されます。このオプションを選択すると、アプリケーションはオンプレミストラフィックのインターフェイスエンドポイントのみを使用します。この VPC 内トラフィックは、低コストのゲートウェイエンドポイントを自動的に使用します。または、[インバウンドエンドポイントのみプライベート DNS を有効にする] をオフにして、すべての S3 リクエストをインターフェイスエンドポイントにルーティングすることもできます。

AWS CLI の使用

`PrivateDnsOnlyForInboundResolverEndpoint` の値を指定しない場合は、デフォルトで `true` になります。ただし、VPC は設定を適用する前に、VPC にゲートウェイエンドポイントが存在することを確認します。ゲートウェイエンドポイントが VPC に存在する場合、呼び出しは成功します。そうでない場合は、以下のエラーメッセージが表示されます。

PrivateDnsOnlyForInboundResolverEndpoint を true に設定するには、VPC *vpce_id* にサービスのゲートウェイエンドポイントが必要です。

新しい VPC インターフェイスエンドポイントの場合

`private-dns-enabled` および `dns-options` 属性を使用して、コマンドラインからプライベート DNS を有効にします。`dns-options` 属性の `PrivateDnsOnlyForInboundResolverEndpoint` オプションは `true` に設定する必要があります。*`user input placeholders`* を、ユーザー自身の情報に置き換えます。

```
aws ec2 create-vpc-endpoint \
--region us-east-1 \
--service-name s3-service-name \
--vpc-id client-vpc-id \
--subnet-ids client-subnet-id \
--vpc-endpoint-type Interface  \
--private-dns-enabled  \
```

```
--ip-address-type ip-address-type \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=true \
--security-group-ids client-sg-id
```

既存の VPC エンドポイントの場合

既存の VPC エンドポイントにプライベート DNS を使用する場合は、以下のコマンド例を使用して、*user input placeholders* を独自の情報に置き換えます。

```
aws ec2 modify-vpc-endpoint \
--region us-east-1 \
--vpc-endpoint-id client-vpc-id \
--private-dns-enabled \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=false
```

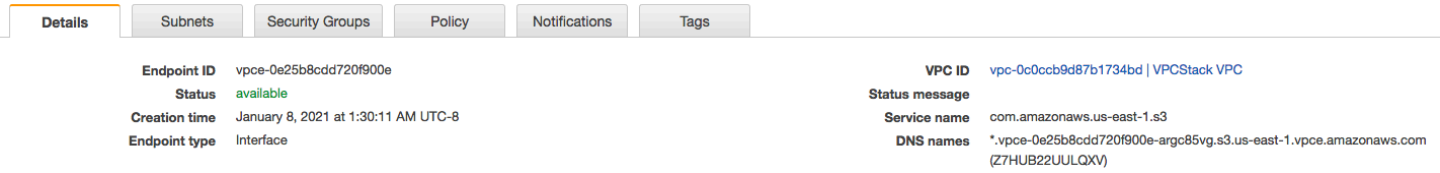
既存の VPC エンドポイントを更新して Inbound Resolver のプライベート DNS のみを有効にする場合は、次の例を使用して、サンプル値を独自の値に置き換えてください。

```
aws ec2 modify-vpc-endpoint \
--region us-east-1 \
--vpc-endpoint-id client-vpc-id \
--private-dns-enabled \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=true
```

S3 インターフェイスエンドポイントからバケット、アクセスポイント、および Amazon S3 コントロール API オペレーションにアクセスする

AWS CLI または AWS SDK を使用し、S3 インターフェイスエンドポイントを介してバケット、S3 アクセスポイント、および Amazon S3 コントロール API オペレーションにアクセスできます。

次の図は、VPC コンソールの [Details] (詳細) タブを示しています。ここでは、VPC エンドポイントの DNS 名を確認できます。この例では、VPC エンドポイント ID (vpce-id) は `vpce-0e25b8cdd720f900e` で、DNS 名は `*.vpce-0e25b8cdd720f900e-argc85vg.s3.us-east-1.vpce.amazonaws.com` です。



DNS 名を使用してリソースにアクセスする場合は、* を適切な値に置き換えてください。* の代わりに使用する適切な値は次のとおりです。

- `bucket`
- `accesspoint`
- `control`

例えば、バケットにアクセスするには、次のような DNS 名を使用します。

`bucket.vpce-0e25b8cdd720f900e-argc85vg.s3.us-east-1.vpce.amazonaws.com`

DNS 名を使用してバケット、アクセスポイント、Amazon S3 Control API オペレーションにアクセスする方法の例については、AWS CLI の例 および AWS SDK の例 の以下のセクションを参照してください。

エンドポイント固有の DNS 名の表示方法の詳細については、VPC ユーザーガイドの「Viewing endpoint service private DNS name configuration」(エンドポイントサービスのプライベート DNS 名設定の表示) を参照してください。

AWS CLI の例

AWS CLI コマンドで S3 インターフェイスエンドポイントを介して S3 バケット、S3 アクセスポイント、または Amazon S3 コントロール API 操作にアクセスするには、`--region` および `--endpoint-url` パラメータを使用します。

例: エンドポイント URL を使用したバケット内のオブジェクトのリスト化

次の例では、バケット名 *`my-bucket`*、リージョン *`us-east-1`*、VPC エンドポイント ID の DNS 名 *`vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com`* をユーザー自身の情報に置き換えます。

```
aws s3 ls s3://my-bucket/ --region us-east-1 --endpoint-url
 https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com
```

例: エンドポイント URL を使用したアクセスポイントのオブジェクトのリスト化

- 方法 1 — アクセスポイントエンドポイントでアクセスポイントの Amazon リソースネーム (ARN) を使用する

  ARN *us-east-1:123456789012:accesspoint/accesspointexamplename*、リージョン *us-east-1*、VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* をユーザー自身の情報に置き換えます。

  ```
  aws s3api list-objects-v2 --bucket arn:aws:s3:us-east-1:123456789012:accesspoint/
  accesspointexamplename --region us-east-1 --endpoint-url
   https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com
  ```

  コマンドを正常に実行できない場合は、AWS CLI を最新バージョンに更新してからやり直してください。更新手順について詳細は、AWS Command Line Interface ユーザーガイドの「AWS CLI の最新バージョンをインストールまたは更新」を参照してください。

- 方法 2 — リージョンバケットエンドポイントでアクセスポイントのエイリアスを使用する

  次の例では、アクセスポイントエイリアス *accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias*、リージョン *us-east-1*、および VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* をユーザー自身の情報に置き換えます。

  ```
  aws s3api list-objects-v2 --
  bucket accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias
   --region us-east-1 --endpoint-url https://bucket.vpce-1a2b3c4d-5e6f.s3.us-
  east-1.vpce.amazonaws.com
  ```

- 方法 3 — アクセスポイントのエイリアスをアクセスポイントエンドポイントで使用する

  まず、バケットをホスト名の一部として含む S3 エンドポイントを作成するには、`aws s3api` が使用するアドレス指定スタイルを `virtual` に設定します。`AWS configure` の詳細については、AWS Command Line Interface ユーザーガイドの「設定ファイルと認証情報ファイルの設定」を参照してください。

  ```
  aws configure set default.s3.addressing_style virtual
  ```

  次に、次の例では、アクセスポイントのエイリアス *accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias*、リージョン *us-east-1*、VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* をユーザー自身の情報に置き換えます。アクセスポイントエイ

リアスの詳細については、「S3 バケットアクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

```
aws s3api list-objects-v2 --
bucket accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias --
region us-east-1 --endpoint-url https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com
```

例: エンドポイント URL を使用して、S3 コントロール API オペレーションでジョブを一覧表示する

次の例では、リージョン *us-east-1*、VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*、およびアカウント ID *12345678* をユーザー自身の情報に置き換えます。

```
aws s3control --region us-east-1 --endpoint-url
 https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com list-jobs --
account-id 12345678
```

AWS SDK の例

AWS SDK を使用する際に S3 インターフェイス エンドポイントを介して S3 バケット、S3 アクセスポイント、または Amazon S3 コントロール API オペレーションにアクセスするには、SDK を最新バージョンに更新します。S3 インターフェイスエンドポイントを介してバケット、アクセスポイント、または S3 コントロール API オペレーションにアクセスするためのエンドポイント URL を使用するようにクライアントを設定します。

SDK for Python (Boto3)

例: エンドポイント URL を使用して S3 バケットにアクセスする

次の例では、リージョン *us-east-1* および VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* をユーザー自身の情報に置き換えます。

```
s3_client = session.client(
service_name='s3',
region_name='us-east-1',
endpoint_url='https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com'
)
```

例: エンドポイント URL を使用して S3 アクセスポイントにアクセスする

次の例では、リージョン *us-east-1* および VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* をユーザー自身の情報に置き換えます。

```
ap_client = session.client(
service_name='s3',
region_name='us-east-1',
endpoint_url='https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com'
)
```

例: エンドポイント URL を使用して Amazon S3 コントロール API にアクセスする

次の例では、リージョン *us-east-1* および VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* をユーザー自身の情報に置き換えます。

```
control_client = session.client(
service_name='s3control',
region_name='us-east-1',
endpoint_url='https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com'
)
```

SDK for Java 1.x

例: エンドポイント URL を使用して S3 バケットにアクセスする

次の例では、VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* をユーザー自身の情報に置き換えます。

```
// bucket client
final AmazonS3 s3 = AmazonS3ClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
List<Bucket> buckets = s3.listBuckets();
```

例: エンドポイント URL を使用して S3 アクセスポイントにアクセスする

次の例では、VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* と ARN *us-east-1:123456789012:accesspoint/prod* をユーザー自身の情報に置き換えます。

```
// accesspoint client
final AmazonS3 s3accesspoint =
 AmazonS3ClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
ObjectListing objects = s3accesspoint.listObjects("arn:aws:s3:us-east-1:123456789012:accesspoint/prod");
```

例: エンドポイント URL を使用して Amazon S3 コントロール API オペレーションにアクセスする

次の例では、VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* をユーザー自身の情報に置き換えます。

```
// control client
final AWSS3Control s3control =
 AWSS3ControlClient.builder().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
final ListJobsResult jobs = s3control.listJobs(new ListJobsRequest());
```

SDK for Java 2.x

例: エンドポイント URL を使用して S3 バケットにアクセスする

次の例では、VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* とリージョン *Region.US_EAST_1* をユーザー自身の情報に置き換えます。

```
// bucket client
Region region = Region.US_EAST_1;
s3Client = S3Client.builder().region(region)

 .endpointOverride(URI.create("https://bucket.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com"))
                    .build()
```

例: エンドポイント URL を使用して S3 アクセスポイントにアクセスする

次の例では、VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* とリージョン *Region.US_EAST_1* をユーザー自身の情報に置き換えます。

```
// accesspoint client
Region region = Region.US_EAST_1;
s3Client = S3Client.builder().region(region)

 .endpointOverride(URI.create("https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com"))
                    .build()
```

例: エンドポイント URL を使用して Amazon S3 コントロール API にアクセスする

次の例では、VPC エンドポイント ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* とリージョン *Region.US_EAST_1* をユーザー自身の情報に置き換えます。

```
// control client
Region region = Region.US_EAST_1;
s3ControlClient = S3ControlClient.builder().region(region)

 .endpointOverride(URI.create("https://control.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com"))
                                  .build()
```

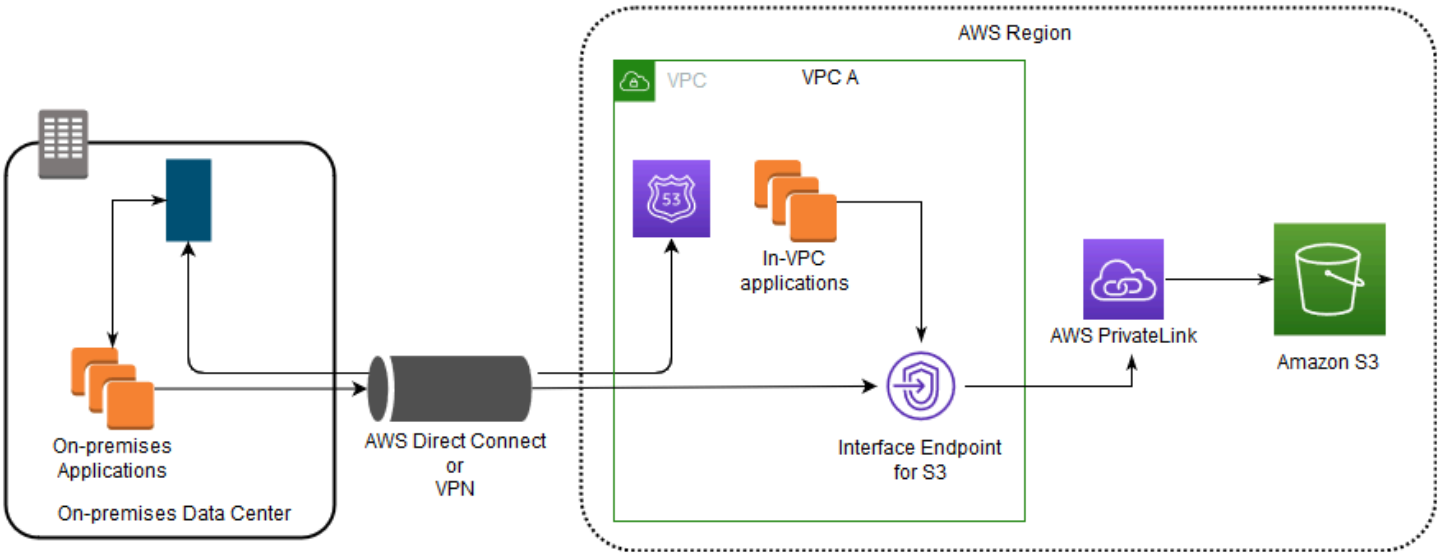
オンプレミスの DNS 設定の更新

エンドポイント固有の DNS 名を使用して Amazon S3 のインターフェイスエンドポイントにアクセスする場合、オンプレミス DNS リゾルバーを更新する必要はありません。パブリック Amazon S3

DNS ドメインからのインターフェイスエンドポイントのプライベート IP アドレスを使用して、エンドポイント固有の DNS 名を解決できます。

Amazon S3 にアクセスするためのインターフェイスエンドポイントの使用 (VPC 内のゲートウェイエンドポイントまたはインターネットゲートウェイの使用なし)

次の図に示すように、VPC 内のインターフェイスエンドポイントは、VPC 内アプリケーションとオンプレミスアプリケーションの両方を Amazon ネットワーク経由で Amazon S3 にルーティングできます。

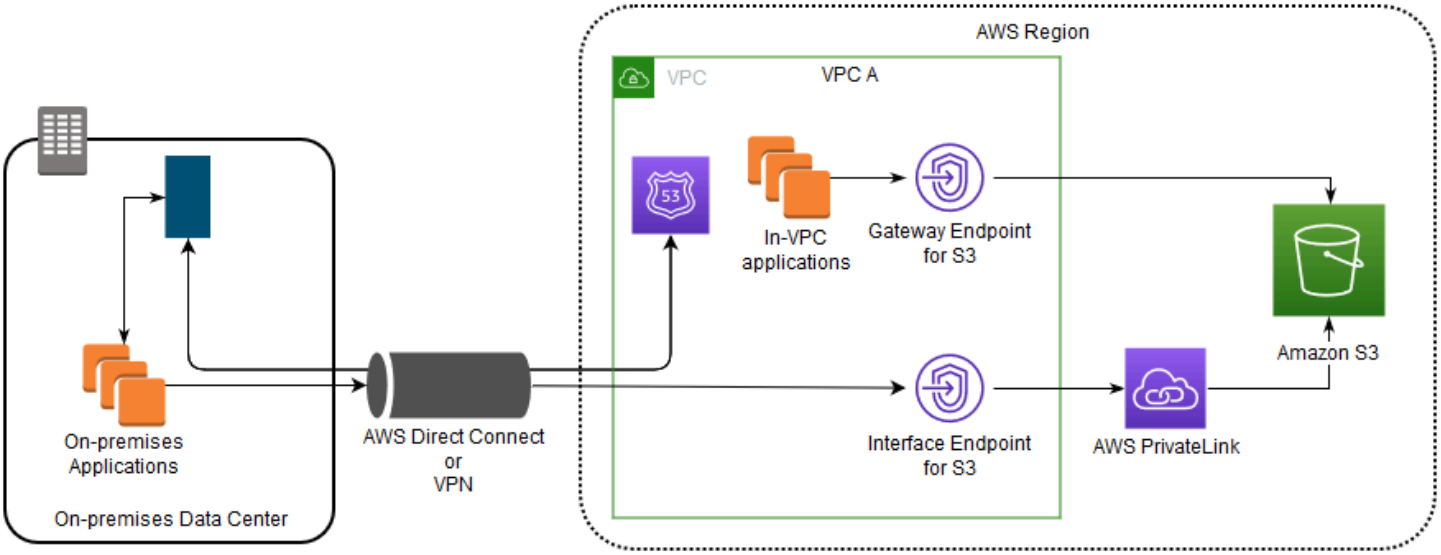


図は、以下を示しています。

- オンプレミスネットワークでは、AWS Direct Connect または AWS VPN を使用して VPC A に接続します。
- オンプレミスと VPC A のアプリケーションでは、エンドポイント固有の DNS 名を使用して S3 インターフェイスエンドポイントを介して Amazon S3 にアクセスします。
- オンプレミスのアプリケーションは、AWS Direct Connect (または AWS VPN) を介して VPC 内のインターフェイスエンドポイントにデータを送信します。AWS PrivateLink は、AWS ネットワークを経由してデータをインターフェイスエンドポイントから Amazon S3 に移動します。
- VPC 内アプリケーションは、インターフェイスエンドポイントにトラフィックの送信も行います。AWS PrivateLink は、AWS ネットワークを経由してデータをインターフェイスエンドポイントから Amazon S3 に移動します。

Amazon S3 にアクセスするための、ゲートウェイエンドポイントとインターフェイスエンドポイントの同じ VPC における併用

次の図に示すように、インターフェイスエンドポイントを作成し、同じ VPC 内に既存のゲートウェイエンドポイントを保持できます。このアプローチにより、VPC 内アプリケーションがゲートウェイエンドポイントを介して Amazon S3 に引き続きアクセスすることを許可します。これについての請求はありません。その後、オンプレミスのアプリケーションだけがインターフェイスエンドポイントを使用して Amazon S3 にアクセスします。この方法で Amazon S3 にアクセスするには、Amazon S3 のエンドポイント固有の DNS 名を使用するようにオンプレミスのアプリケーションを更新する必要があります。



図は、以下を示しています。

- オンプレミスのアプリケーションは、エンドポイント固有の DNS 名を使用し、AWS Direct Connect (または AWS VPN) を介して VPC 内のインターフェイスエンドポイントにデータを送信します。AWS PrivateLink は、AWS ネットワークを経由してデータをインターフェイスエンドポイントから Amazon S3 に移動します。
- VPC 内アプリケーションは、デフォルトのリージョンの Amazon S3 名を使用し、AWS ネットワークを介して Amazon S3 に接続するゲートウェイエンドポイントにデータを送信します。

ゲートウェイエンドポイントの詳細については、VPC ユーザーガイドのゲートウェイ VPC エンドポイントを参照してください。

Amazon S3 用の VPC エンドポイントポリシーの作成

VPC エンドポイントに Amazon S3 へのアクセスをコントロールするエンドポイントポリシーをアタッチできます。このポリシーでは、以下の情報を指定します。

- アクションを実行できる AWS Identity and Access Management (IAM) プリンシパル
- 実行可能なアクション
- アクションを実行できるリソース

Amazon S3 バケットポリシーを使用して、バケットポリシーの `aws:sourceVpce` 条件を使用して、特定の VPC エンドポイントからの特定のバケットへのアクセスを制限することもできます。次の例は、バケットまたはエンドポイントへのアクセスを制限するポリシーを示しています。

トピック

例: VPC エンドポイントから特定のバケットへのアクセスの制限

特定の Amazon S3 バケットへのアクセスのみを制限するエンドポイントポリシーを作成できます。このタイプのポリシーは、VPC で S3 バケットを使用する他の AWS のサービス がある場合に便利です。次のバケットポリシーは、*example-s3-bucket1* へのアクセスのみを制限します。このエンドポイントポリシーを使用するには、*example-s3-bucket1* をバケットの名前に置き換えます。

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909151",
  "Statement": [
    { "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::example-s3-bucket1",
                   "arn:aws:s3:::example-s3-bucket1/*"]
```

```
    }
  ]
}
```

例: VPC エンドポイントから特定のアカウントのバケットへのアクセスの制限

特定の AWS アカウント の S3 バケットへのアクセスのみを制限するエンドポイントポリシーを作成できます。VPC 内のクライアントが所有していないバケットにアクセスできないようにするには、エンドポイント ポリシーで次のステートメントを使用してください。次のステートメント例では、単一の AWS アカウント、*111122223333* が所有するリソースへのアクセスを制限するポリシーを作成します。

```
{
  "Statement": [
    {
      "Sid": "Access-to-bucket-in-specific-account-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Deny",
      "Resource": "arn:aws:s3:::*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

> **Note**
>
> アクセスされるリソースの AWS アカウント ID を指定するには、IAM ポリシーで `aws:ResourceAccount` キーまたは `s3:ResourceAccount` キーを使用できます。ただし、一部の AWS のサービス は AWS マネージドバケットへのアクセスに依存していることに注意してください。したがって、IAM ポリシーで `aws:ResourceAccount` キーまたは `s3:ResourceAccount` キーを使用することは、これらのリソースへのアクセスにも影響する可能性があります。

例: S3 バケットポリシーでの特定の VPC エンドポイントへのアクセスの制限

例: S3 バケットポリシーでの特定の VPC エンドポイントへのアクセスの制限

次の Amazon S3 バケットポリシーは、VPC エンドポイント *vpce-1a2b3c4d* からのみ、特定のバケット、*example-s3-bucket2* へのアクセスを許可します。指定されたエンドポイントを使用していない場合、ポリシーによりバケットへのすべてのアクセスが拒否されます。aws:sourceVpce 条件はエンドポイントを指定し、VPC エンドポイントリソースの Amazon リソースネーム (ARN) を必要とせず、エンドポイント ID のみを指定します。このバケットポリシーを使用するには、*example-s3-bucket2* と *vpce-1a2b3c4d* をバケット名とエンドポイントに置き換えてください。

Important

- 次の Amazon S3 バケットポリシーを適用して特定の VPC エンドポイントのみへのアクセスを制限すると、バケットへのアクセスが意図せずにブロックされる場合があります。特に VPC エンドポイントからの接続に対するバケットのアクセスを制限することを目的としたバケットのポリシーにより、バケットへのすべての接続がブロックされる場合があります。この問題を修正する方法については、「バケットポリシーの VPC または VPC エンドポイント ID が間違っています。ポリシーを修正してバケットにアクセスできるようにするにはどうすれば良いですか?」(AWS Support ナレッジセンター) を参照してください。
- 次のポリシーの例を使用する前に、VPC エンドポイントの ID をユースケースに応じた値に置き換えてください。そうしないと、バケットにアクセスできません。
- このポリシーは、コンソールリクエストが指定の VPC エンドポイントを経由していない場合、指定先のバケットへのコンソールアクセスを無効にします。

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909152",
  "Statement": [
    { "Sid": "Access-to-specific-VPCE-only",
      "Principal": "*",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::example-s3-bucket2",
                   "arn:aws:s3:::example-s3-bucket2/*"],
      "Condition": {"StringNotEquals": {"aws:sourceVpce": "vpce-1a2b3c4d"}}
```

```
    }
  ]
}
```

ポリシーの例については、VPC ユーザーガイドの Amazon S3 のエンドポイントを参照してください。

VPC 接続の詳細については、 ホワイトペーパー Amazon Virtual Private Cloud 接続オプションAWS のネットワークから VPC への接続オプション を参照してください。

アクセス管理

AWS では、リソースはユーザーが操作できるエンティティです。Amazon Simple Storage Service (S3) では、バケットとオブジェクトが元の Amazon S3 リソースです。すべての S3 のお客様のバケットには、オブジェクトが含まれている可能性があります。新機能が S3 に追加され、リソースもさらに追加されましたが、すべてのお客様がこれらの機能固有のリソースを使用しているわけではありません。Amazon S3 リソースの詳細については、「S3 リソース」を参照してください。

デフォルトでは、すべての Amazon S3 リソースはプライベートです。デフォルトでは、リソースを作成した AWS アカウント のルートユーザー (リソース所有者) と、必要なアクセス許可を持つそのアカウント内の IAM ユーザーは、作成したリソースにアクセスできます。リソース所有者は、リソースにアクセスできる他のユーザーと、他のユーザーがリソースに対して実行できるアクションを決定します。S3 には、S3 リソースへのアクセスを他のユーザーに許可するために使用できるさまざまなアクセス管理ツールがあります。

以下のセクションでは、S3 リソースの概要、使用可能な S3 アクセス管理ツール、各アクセス管理ツールの最適なユースケースについて説明します。これらのセクションのリストは、包括的な内容で、すべての S3 リソース、アクセス管理ツール、一般的なアクセス管理のユースケースが含まれることを意図しています。同時に、これらのセクションは、必要な技術的情報へと案内するディレクトリとなるように設計されています。以下のトピックの一部をよく理解している場合は、該当するセクションに進んでください。

トピック

- S3 リソース
- ID
- アクセス管理ツール
- アクション

- アクセス管理のユースケース
- アクセス管理のトラブルシューティング
- Amazon S3 用 Identity and Access Management
- S3 Access Grants でのアクセス管理
- ACL によるアクセス管理
- Amazon S3 ストレージへのパブリックアクセスのブロック
- IAM Access Analyzer for S3 を使用したバケットアクセスの確認
- バケット所有者条件によるバケット所有者の確認
- オブジェクトの所有権の制御とバケットの ACL の無効化。

S3 リソース

元の Amazon S3 リソースとは、バケットとそれらに含まれるオブジェクトです。S3 に新しい機能が追加されると、新しいリソースも追加されます。以下は、S3 リソースとそれぞれの特徴の完全なリストです。

リソースタイプ	Amazon S3 の特徴	説明
`bucket`	主要機能	バケット とは、オブジェクトのコンテナのことです。S3 にオブジェクトを保存するには、バケットを作成してから、1 つまたは複数のオブジェクトをバケットにアップロードします。詳細については、「Amazon S3 バケットの作成、設定、操作」を参照してください。
`object`		オブジェクトには、ファイルと、そのファイルを記述している任意のメタデータを指定できます。オブジェクトがバケットの中にあるときは、オブジェクトを開き、ダウンロードして、移動させます。詳細については、「Amazon S3 でのオブジェクトのアップロード、ダウンロード、操作」を参照してください。
`accesspoint`	アクセスポイント	Access Points は、バケットにアタッチされた名前付きのネットワークエンドポイントで、Amazon S3 オブジェクトのオペレーション (`GetObject` や `PutObject` など)

リソースタイプ	Amazon S3 の特徴	説明
		を実行するために使用できます。各アクセスポイントには、個別の許可とネットワーク制御、および基になるバケットにアタッチされたバケットポリシーと連動して機能するカスタマイズされたアクセスポイントポリシーがあります。仮想プライベートクラウド (VPC) からのリクエストのみを受け入れるように任意のアクセスポイントを設定したり、アクセスポイントごとにカスタムブロックパブリックアクセスを設定したりできます。詳細については、「Amazon S3 アクセスポイントを使用したデータアクセスの管理」を参照してください。
`objectlambdaaccesspoint`		Object Lambda アクセスポイントは、Lambda 関数にも関連付けられているバケットのアクセスポイントです。Object Lambda アクセスポイントを使用すると、Amazon S3 `GET`、`LIST`、`HEAD` リクエストに独自のコードを追加して、データがアプリケーションに返されるときにそのデータを変更および処理できます。詳細については、「Object Lambda アクセスポイントの作成」を参照してください。

リソースタイプ	Amazon S3 の特徴	説明
`multiregionaccesspoint`		マルチリージョン Access Points を使用すると、アプリケーションが複数の AWS リージョンにある Amazon S3 バケットからのリクエストを実行するために使用できるグローバルエンドポイントを作成できます。マルチリージョンアクセスポイントを使用して、単一のリージョンで使用するのと同じシンプルなアーキテクチャでマルチリージョンアプリケーションを構築し、世界中のどこでもこれらのアプリケーションを実行することができます。混雑したパブリックインターネット経由でリクエストを送信する代わりに、マルチリージョンアクセスポイントのグローバルエンドポイントに対して行われたアプリケーションリクエストは、AWS グローバルネットワークを介して最も近い Amazon S3 バケットに自動的にルーティングされます。詳細については、「Amazon S3 マルチリージョンアクセスポイント」を参照してください。
`job`	S3 バッチオペレーション	ジョブは S3 バッチオペレーション機能のリソースです。S3 バッチオペレーションを使用すると、指定した Amazon S3 のオブジェクトのリストに対して大規模なバッチオペレーションを実行できます。Amazon S3 は、バッチオペレーションジョブの進捗状況の追跡、通知の送信、すべてのアクションの詳細な完了レポートの保存を行い、フルマネージド型の監査可能なサーバーレスエクスペリエンスを提供します。詳細については、「Amazon S3 オブジェクトでの大規模なバッチ操作の実行」を参照してください。

リソースタイプ	Amazon S3 の特徴	説明
`storagelensconfiguration`	S3 Storage Lens	S3 Storage Lens は、組織全体のストレージメトリクスとユーザーデータを収集します。S3 Storage Lens は、組織内の数百または数千のアカウントのオブジェクトストレージの使用状況とアクティビティを 1 つのビューで表示し、複数の集約レベルでインサイトを生成するための詳細情報を提供します。詳細については、「Amazon S3 ストレージレンズを使用してストレージのアクティビティと使用状況を評価する」を参照してください。
`storagelensgroup`		S3 Storage Lens グループは、オブジェクトメタデータに基づくカスタムフィルターを使用してメトリクスを集約します。S3 Storage Lens グループを使用すると、経過時間別のオブジェクトの分布や最も一般的なファイルタイプなど、データの特性を調べることができます。詳細については、「S3 Storage Lens グループの使用」を参照してください。
`accessgrantsinstance`	S3 Access Grants	S3 Access Grants インスタンスは、ユーザーが作成した S3 アクセス許可のコンテナです。S3 Access Grants を使用すると、 アカウント内の IAM アイデンティティ、他の アカウントの IAM アイデンティティ (クロスアカウント) 、および社内ディレクトリから AWS IAM Identity Center に追加されたディレクトリアイデンティティの Amazon S3 データへのアクセス許可を作成できます。S3 Access Grants の詳細については、「S3 Access Grants でのアクセス管理」を参照してください。

リソースタイプ	Amazon S3 の特徴	説明
`accessgrantslocation`		Access Grants Location は、S3 Access Grants インスタンスに登録するバケット、バケット内のプレフィックス、またはオブジェクトです。S3 Access Grants インスタンス内に口ケーションを登録してから、その口ケーションへのアクセス許可を作成する必要があります。次に、S3 Access Grants を使用して、 アカウント内の IAM アイデンティティ、他のアカウントの IAM アイデンティティ (クロスアカウント) 、および社内ディレクトリから AWS IAM Identity Center に追加されたディレクトリアイデンティティのバケット、プレフィックス、またはオブジェクトへのアクセス許可を作成できます。S3 Access Grants の詳細については、「S3 Access Grants でのアクセス管理」を参照してください。
`accessgrant`		Access Grant は、Amazon S3 データに対する個別のアクセス許可です。S3 Access Grants を使用すると、 アカウント内の IAM アイデンティティ、他の アカウントの IAM アイデンティティ (クロスアカウント) 、および社内ディレクトリから AWS IAM Identity Center に追加されたディレクトリアイデンティティの Amazon S3 データへのアクセス許可を作成できます。S3 Access Grants の詳細については、「S3 Access Grants でのアクセス管理」を参照してください。

バケット

Amazon S3 バケットには、汎用バケットとディレクトリバケットの 2 種類があります。

- 汎用バケット はオリジナルの S3 バケットタイプであり、ほとんどのユースケースやアクセスパターンに推奨されます。汎用バケットでは、S3 Express One Zone 以外のすべてのストレージクラスにオブジェクトを保存することもできます。S3 ストレージクラスの詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。
- ディレクトリバケットは、S3 Express One Zone ストレージクラスのみを使用します。アプリケーションがパフォーマンスの影響を受けやすく、1 桁ミリ秒の PUT と GET のレイテンシーから利点が得られる場合にお勧めします。詳細については、ディレクトリバケット、S3 Express One

Zone とは、およびS3 Express One Zone 向け AWS Identity and Access Management (IAM)を参照してください。

S3 リソースの分類

Amazon S3 には、S3 リソースを分類および整理する機能が備えられています。リソースの分類は、リソースの整理に役立つだけでなく、これにより、リソースカテゴリに基づいてアクセス管理ルールを設定することもできます。特に、プレフィックスとタグ付けは、アクセス管理のアクセス許可を設定するときに使用する 2 つのストレージ組織機能です。

> **Note**
>
> 以下の情報は、汎用バケットに適用されます。ディレクトリバケットはタグ付けをサポートしておらず、プレフィックスの誓約があります。詳細については、「S3 Express One Zone 向け AWS Identity and Access Management (IAM)」を参照してください。

- プレフィックス — Amazon S3 のプレフィックスは、S3 バケットに保存されたオブジェクトを整理するために使用されるオブジェクトキー名の先頭にある文字列です。スラッシュ (/) などの区切り文字を使用して、オブジェクトキー名内のプレフィックスの末尾を指定できます。例えば、`engineering/` プレフィックスで始まるオブジェクトキー名や、`marketing/campaigns/` プレフィックスで始まるオブジェクトキー名があるとします。スラッシュ文字 (/) など、プレフィックスの末尾に区切り記号を使用すると、フォルダとファイルの命名規則がエミュレートされます。ただし、S3 では、プレフィックスはオブジェクトキー名の一部です。汎用 S3 バケットでは、実際のフォルダ階層は存在しません。

  Amazon S3 では、プレフィックスを使用してオブジェクトを整理およびグループ化できます。オブジェクトへのアクセスをプレフィックスで管理することもできます。例えば、特定のプレフィックスで始まる名前のオブジェクトへのアクセスのみを制限できます。

  詳細については、「プレフィックスを使用してオブジェクトを整理する」を参照してください。S3 コンソールは、フォルダの概念を使用します。これは、汎用バケットでは、基本的にオブジェクトキー名の前に付加されるプレフィックスです。詳細については、「フォルダを使用して Amazon S3 コンソールのオブジェクトを整理する」を参照してください。
- タグ — 各タグは、リソースに割り当てることができるキーと値のペアです。例えば、一部のリソースにタグ `topicCategory=engineering` を付けることができます。タグ付けを使用すると、コスト配分、分類と整理、アクセスコントロールに役立ちます。バケットのタグ付けはコスト配分にのみ使用されます。オブジェクト、S3 Storage Lens、ジョブ、および S3 Access Grants に

は、整理またはアクセスコントロールの目的でタグ付けできます。S3 Access Grants では、コスト配分にもタグ付けを使用できます。タグを使用してリソースへのアクセスを制御する例として、特定のタグまたはタグの組み合わせを持つオブジェクトのみを共有できます。

詳細については、IAM ユーザーガイドの「リソースタグを使用した AWS リソースへのアクセスの制御」を参照してください。

ID

Amazon S3 では、リソース所有者は、バケットやオブジェクトなどのリソースを作成した ID です。デフォルトでは、必要なアクセス許可を持つアカウント内でリソースと IAM アイデンティティを作成したアカウントのルートユーザーのみが S3 リソースにアクセスできます。リソース所有者は、他のアイデンティティに S3 リソースへのアクセスを許可できます。

リソースを所有していないアイデンティティは、そのリソースへのアクセスをリクエストできます。リソースに対するリクエストは、認証済みまたは未認証です。認証リクエストには、リクエストの送信元を認証する署名値を含める必要がありますが、未認証のリクエストでは署名は不要です。認証されたユーザーのみにアクセスを許可することをお勧めします。リクエストの認証の詳細については、リクエストの実行 を参照してください。

> **Important**
>
> 認証されたリクエストの実行に AWS アカウントのルートユーザー認証情報を使用しないことをお勧めします。代わりに、IAM ロールを作成し、このロールにフルアクセスを許可します。このロールを持つユーザーを管理者ユーザーと呼びます。AWS アカウントのルートユーザー認証情報ではなく、管理者ユーザーに割り当てられた認証情報を使用して AWS とやり取りし、バケットの作成、ユーザーの作成、アクセス許可の付与などのタスクを実行できます。詳細については、AWS 全般のリファレンスの「AWS アカウント ルートユーザーの認証情報と IAM ユーザーの認証情報」および 「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

Amazon S3 のデータにアクセスするアイデンティティは、次のいずれかになります。

AWS アカウント所有者

リソースを作成した AWS アカウントです。例えば、バケットを作成したアカウントです。このアカウントは リソースを所有しています。詳細については、「AWS アカウントのルートユーザー」を参照してください。

AWS アカウント所有者の同じアカウントの IAM アイデンティティ

S3 アクセスを必要とする新しいチームメンバーのアカウントを設定する場合、AWS アカウント所有者は AWS Identity and Access Management (IAM) を使用してユーザー、グループ、ロールを作成できます。その後、AWS アカウント所有者はこれらの IAM アイデンティティとリソースを共有できます。アカウント所有者は、IAM アイデンティティに付与するアクセス許可を指定することもできます。これにより、共有リソースで実行できるアクションを許可または拒否できます。

IAM アイデンティティを使用すると、共有リソースにアクセスする前にログイン認証情報の入力を要求するなど、機能を強化できます。IAM アイデンティティを使用すると、強力なアイデンティティ基盤をサポートするために、IAM の多要素認証 (MFA) 形式を実装することもできます。IAM のベストプラクティスは、個々のユーザーにアクセス許可を付与するのではなく、アクセス管理用のロールを作成することです。個々のユーザーを適切なロールに割り当てます。詳細については、「IAM のセキュリティのベストプラクティス」を参照してください。

他の AWS アカウント所有者とその IAM アイデンティティ (クロスアカウントアクセス)

AWS アカウント所有者は、他の AWS アカウント所有者、または別の AWS アカウントに属する IAM アイデンティティにリソースへのアクセスを許可することもできます。

> **Note**
>
> アクセス許可の委任 — AWS アカウントがリソースを所有している場合、そのアカウントは別の AWS アカウントにこれらの許可を付与できます。そのアカウントは、それらのアクセス許可またはそのサブセットを、同じアカウント内のユーザーに委任できます。これはアクセス許可の委任と呼ばれます。ただし、他のアカウントから許可を受け取るアカウントは、別の AWS アカウントに「クロスアカウント」でこれらの許可を委任することはできません。

匿名ユーザー (パブリックアクセス)

AWS アカウント 所有者はリソースを公開できます。リソースを公開すると、リソースが匿名ユーザーと技術的に共有されます。2023 年 4 月以降に作成されたバケットは、この設定を変更しない限り、デフォルトですべてのパブリックアクセスをブロックします。パブリックアクセスをブロックするようにバケットを設定し、認証されたユーザーにのみアクセスを許可することをお勧めします。ブロックパブリックアクセスの詳細については、Amazon S3 ストレージへのパブリックアクセスのブロック を参照してください。

AWS のサービス

リソース所有者は、Amazon S3 リソースへの別の AWS サービスアクセスを許可できます。例えば、ログファイルをバケットに書き込む `s3:PutObject` アクセス許可を AWS CloudTrail サービスに付与できます。詳細については、「AWS サービスへのアクセスの提供」を参照してください。

社内ディレクトリのアイデンティティ

リソース所有者は、S3 Access Grants を使用して、社内ディレクトリのユーザーまたはロールに S3 リソースへのアクセス権を付与できます。AWS IAM Identity Center への社内ディレクトリの追加に関する詳細は、「IAM Identity Center とは何ですか?」を参照してください。

バケットまたはリソース所有者

このようなリソースの所有者は、バケットの作成とオブジェクトのアップロードに使用する AWS アカウントです。バケット所有者は、別の AWS アカウント (または別のアカウントのユーザー) に対して、オブジェクトをアップロードするためのクロスアカウントアクセス許可を付与できます。

バケット所有者が別のアカウントにバケットへのオブジェクトのアップロードを許可すると、バケット所有者はデフォルトで、バケットにアップロードされたすべてのオブジェクトを所有します。ただし、バケット所有者の強制バケット設定とバケット所有者の優先バケット設定の両方がオフになっている場合、オブジェクトをアップロードする AWS アカウントがこれらのオブジェクトを所有し、バケット所有者には別のアカウントが所有するオブジェクトに対するアクセス許可は付与されません。ただし、以下の例外があります。

- バケット所有者が請求の支払いを行う場合、バケット所有者は、オブジェクトの所有者に関係なく、オブジェクトへのアクセスを拒否したり、バケット内のオブジェクトを削除したりすることができます。
- バケット所有者は、オブジェクトの所有者に関係なく、オブジェクトをアーカイブしたり、アーカイブされたオブジェクトを復元したりすることができます。アーカイブはオブジェクトの格納に使用されるストレージクラスを指します。詳細については、「ストレージのライフサイクルの管理」を参照してください。

アクセス管理ツール

Amazon S3 には、さまざまなセキュリティ機能とツールが用意されています。以下は、これらの機能とツールの包括的なリストです。これらのアクセス管理ツールがすべて必要なわけではありませんが、Amazon S3 リソースへのアクセスを付与するには、これらのツールを 1 つ以上使用する必要があります。これらのツールを適切に適用すると、データの整合性を維持し、目的のユーザーがリソースにアクセスできるようにするために役立ちます。

最も一般的に使用されるアクセス管理ツールは、アクセスポリシーです。アクセスポリシーは、バケットのバケットポリシーなど、AWS リソースにアタッチされたリソースベースのポリシーに指定できます。アクセスポリシーは、IAM ユーザー、グループ、ロールなどの AWS Identity and Access Management (IAM) アイデンティティにアタッチされたアイデンティティベースのポリシーに指定することもできます。アクセスポリシーを作成して、 AWS アカウントおよび IAM ユーザー、グループ、ロールにアクセス許可を付与し、リソースに対してオペレーションを実行します。例えば、他のアカウントがバケットにオブジェクトをアップロードできるように、別の AWS アカウントに PUT Object アクセス許可を付与できます。

アクセスポリシーでは、誰が何にアクセスできるかを記述します。Amazon S3 は、リクエストを受け取ったときにすべてのアクセスポリシーを評価して、リクエストを許可するか拒否するかを判断する必要があります。Amazon S3 がこれらのポリシーを評価する方法の詳細については、Amazon S3 がリクエストを許可する仕組み を参照してください。

Amazon S3 で使用できるアクセス管理ツールを以下に示します。

バケットポリシー

Amazon S3 バケットポリシーは、特定のバケットにアタッチされた JSON 形式の AWS Identity and Access Management (IAM) リソースベースのポリシーです。バケットポリシーを使用すると、バケットおよびバケット内のオブジェクトに対するアクセス許可を、他の AWS アカウントまたは IAM アイデンティティに付与できます。S3 アクセス管理の多くのユースケースは、バケットポリシーを使用することで要件が満たされます。バケットポリシーを使用すると、承認されたアイデンティティのみがリソースにアクセスしてアクションを実行できるように、バケットアクセスをパーソナライズすることができます。詳細については、「Amazon S3 のバケットポリシー」を参照してください。

以下は、バケットポリシーの例です。JSON ファイルを使用してバケットポリシーを表現します。このポリシーの例では、バケット内のすべてのオブジェクトに IAM ロールの読み取りアクセス許可を付与します。このポリシーには、DOC-EXAMPLE-BUCKET1 という名前のバケット内のオブジェクトに対して、s3:GetObject アクション (読み取りアクセス許可) を許可する BucketLevelReadPermissions という名前の 1 個のステートメントがあります。IAM ロールを Principal として指定することで、このポリシーは、このロールを持つすべての IAM ユーザーにアクセス権を付与します。このポリシーの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Sid":"BucketLevelReadPermissions",
      "Effect":"Allow",
      "Principal": {
     "AWS": "arn:aws:iam::123456789101:role/s3-role"
      },
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"]
    }]
  }
```

> **Note**
>
> ポリシーを作成するときは、Principal 要素にワイルドカード文字 (*) を使用しないでください。ワイルドカードを使用すると、すべてのユーザーが Amazon S3 リソースにアクセスできるようになります。代わりに、バケットへのアクセスが許可されているユーザーまたはグループを明示的にリストするか、ポリシーで条件句を使用して満たす必要がある条件をリストします。ユーザーまたはグループのアクションにワイルドカード文字を含めるのではなく、該当する場合に特定のアクセス許可を付与します。

ID ベースのポリシー

アイデンティティベースのポリシーまたは IAM ユーザーポリシーは、AWS Identity and Access Management (IAM) ポリシーの一種です。アイデンティティベースのポリシーは、AWSアカウントの IAM ユーザー、グループ、またはロールにアタッチされる JSON 形式のポリシーです。アイデンティティベースのポリシーを使用すると、バケットまたはオブジェクトへのアクセス許可を IAM アイデンティティに付与できます。アカウント内に IAM ユーザー、グループ、およびロールを作成し、これらにアクセスポリシーをアタッチできます。次に、Amazon S3 リソースを含む AWS リソースへのアクセス許可を付与できます。詳細については、「Amazon S3 のアイデンティティベースのポリシー」を参照してください。

以下は、アイデンティティベースポリシーの例です。このポリシーの例では、関連 IAM ロールに、バケットとバケット内のオブジェクトに対して Amazon S3 の 6 つの異なるアクション (アクセス許可) の実行を許可します。このポリシーをアカウントの IAM ロールにアタッチし、そのロールを一部の IAM ユーザーに割り当てると、このロールを持つユーザーは、ポリシーで指定されたリソース (バケット) に対してこれらのアクションを実行できます。このポリシーの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
  {
    "Sid": "AssignARoleActions",
    "Effect": "Allow",
    "Action": [
   "s3:PutObject",
   "s3:GetObject",
   "s3:ListBucket",
   "s3:DeleteObject",
   "s3:GetBucketLocation"
    ],
    "Resource": [
   "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*",
   "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
    ]
    },
  {
    "Sid": "AssignARoleActions2",
    "Effect": "Allow",
    "Action": "s3:ListAllMyBuckets",
    "Resource": "*"
  }
  ]
}
```

S3 Access Grants

S3 Access Grants を使用して、Active Directory などの企業 ID ディレクトリ内のアイデンティティと AWS Identity and Access Management (IAM) アイデンティティの両方の Amazon S3 データへのアクセス許可を作成します。S3 Access Grants は、大規模なデータのアクセス許可の管理に役立ちます。さらに、S3 Access Grants は、エンドユーザーアイデンティティと、AWS CloudTrail での S3 データへのアクセスに使用されるアプリケーションをログに記録します。これにより、S3 バケット内のデータへのすべてのアクセスについて、エンドユーザーアイデンティティまで詳細な監査履歴が提供されます。詳細については、「S3 Access Grants でのアクセス管理」を参照してください。

アクセスポイント

Amazon S3 Access Points は、S3 の共有データセットを使用するアプリケーションの大規模なデータアクセスの管理を簡素化します。Access Points は名前付きのネットワークエンドポイントで、バケットにアタッチされます。アクセスポイントを使用すると、オブジェクトのアップロードや取得などの S3 オブジェクトオペレーションを大規模に実行できます。バケットには最大 10,000 個のア

クセスポイントをアタッチできます。各アクセスポイントでは、個別のアクセス許可やネットワークコントロールを適用することで、S3 オブジェクトへのアクセスをきめ細かく制御できます。S3 Access Points は、同じアカウントまたは別の信頼されたアカウントのバケットに関連付けることができます。Access Points ポリシーは、基になるバケットポリシーと組み合わせて評価されるリソースベースのポリシーです。詳細については、「Amazon S3 アクセスポイントを使用したデータアクセスの管理」を参照してください。

アクセスコントロールリスト (ACL)

ACL は、アクセス許可の被付与者と付与されるアクセス許可を識別するリストです。ACL は、基本的な読み取り/書き込み許可を他の AWS アカウントに付与します。ACL では、Amazon S3 固有の XML スキーマが使用されます。ACL は、AWS Identity and Access Management (IAM) ポリシーの一種です。オブジェクト ACL は、オブジェクトへのアクセスを管理するために使用され、バケット ACL はバケットへのアクセスを管理するために使用されます。バケットポリシーはバケット全体に対して 1 つですが、オブジェクト ACL は各オブジェクトに対して指定されます。オブジェクトごとに個別にアクセスを制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。ACL の使用方法の詳細については、オブジェクトの所有権の制御とバケットの ACL の無効化。 を参照してください。

> **Warning**
>
> Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。

以下に、バケット ACL の例を示します。この ACL アクセス権限は、フルコントロールのアクセス許可を持つバケット所有者を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
 <AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
   <ID>Owner-Canonical-User-ID</ID>
   <DisplayName>owner-display-name</DisplayName>
  </Owner>
 <AccessControlList>
  <Grant>
   <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Canonical
 User">
    <ID>Owner-Canonical-User-ID</ID>
    <DisplayName>display-name</DisplayName>
   </Grantee>
```

```
    <Permission>FULL_CONTROL</Permission>
   </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

オブジェクトの所有権

オブジェクトへのアクセスを管理するには、オブジェクトの所有者である必要があります。オブジェクト所有権のバケットレベルの設定を使用すると、バケットにアップロードされたオブジェクトの所有権を制御できます。また、オブジェクト所有権を使用して、ACL を有効にします。デフォルトでは、オブジェクト所有権は [バケット所有者の強制] に設定され、すべての ACL は無効になっています。ACL が無効になっている場合、バケット所有者はバケット内のすべてのオブジェクトを所有し、データへのアクセスの管理のみを行います。アクセスを管理するために、バケット所有者は ACL を除くポリシーまたは別のアクセス管理ツールを使用します。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

オブジェクト所有権には、バケットにアップロードされるオブジェクトの所有権を制御し、ACL を無効化または有効化するために使用できる 3 つの設定があります。

ACL を無効化する

- バケット所有者強制 (デフォルト) – ACL は無効になり、バケット所有者はバケット内のすべてのオブジェクトを自動的に所有し、完全に制御します。ACL は、S3 バケット内のデータに対するアクセス許可に対して影響を与えません。このバケットはアクセスコントロールを定義するためだけにポリシーを使用します。

ACL を有効化する

- 希望するバケット所有者 — バケット所有者は、他のアカウントが `bucket-owner-full-control` 既定 ACL でバケットに書き込む新しいオブジェクトを所有し、完全にコントロールできます。
- オブジェクトライター — オブジェクトをアップロードする AWS アカウント は、そのオブジェクトを所有し、そのオブジェクトを完全にコントロールし、ACL を介して他のユーザーにそのオブジェクトへのアクセス権を付与できます。

その他のベストプラクティス

転送中および保管時のデータを保護するために、次のバケット設定とツールを使用することを検討してください。どちらも、データの完全性とアクセシビリティを維持するために重要です。

- パブリックアクセスをブロック — デフォルトのバケットレベルの設定の [パブリックアクセスをブロック] をオフにしないでください。この設定は、デフォルトでデータへのパブリックアクセスをブロックします。ブロックパブリックアクセスの詳細については、Amazon S3 ストレージへのパブリックアクセスのブロック を参照してください。
- S3 バージョニング — データ整合性のために、S3 バージョニングバケット設定を実装できます。この設定では、オブジェクトを上書きするのではなく、更新時にオブジェクトに複数のバージョンを持たせます。S3 バージョニングを使用すると、必要に応じて以前のバージョンを保存、取得、復元できます。S3 バージョニングの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。
- S3 Object Lock — S3 Object Lock は、データの整合性を実現するために実装できるもう 1 つの設定です。この機能により、オブジェクトを不変的に保存するための Write-Once-Read-Many (WORM) モデルを実装できます。Object Lock の詳細については、「S3 オブジェクトロックの使用」を参照してください。
- オブジェクト暗号化 — Amazon S3 には、転送中および保管時のデータを保護するために使用できる、複数のオブジェクト暗号化オプションがあります。サーバー側の暗号化では、オブジェクトをデータセンター内のディスクに保存する前に暗号化し、オブジェクトをダウンロードするときに復号します。リクエストが認証され、お客様がアクセス許可を持っていれば、オブジェクトが暗号化されているかどうかに関係なく同じ方法でアクセスできます。詳細については、「サーバー側の暗号化によるデータの保護」を参照してください。S3 は、新しくアップロードされたオブジェクトをデフォルトで暗号化します。詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。クライアント側の暗号化では、Amazon S3 に送信する前にデータを暗号化します。詳細については、「クライアント側の暗号化を使用したデータの保護」を参照してください。
- 署名方法 — 署名バージョン 4 は、HTTP で送信される AWS リクエストに認証情報を追加するプロセスです。セキュリティ対策として、AWS へのほとんどのリクエストは、アクセスキーを使用して署名する必要があります。アクセスキーは、アクセスキー ID とシークレットアクセスキーで構成されます。これらの 2 つのキーは、一般的にセキュリティ認証情報と呼ばれます。詳細については、「リクエストの認証 (AWS 署名バージョン 4)」および「署名バージョン 4 の署名プロセス」を参照してください。

アクション

S3 のアクセス許可および条件キーの完全なリストについては、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

アクション

Amazon S3 の AWS Identity and Access Management (IAM) アクションは、S3 バケットまたはオブジェクトで実行できるアクションです。これらのアクションをアイデンティティに付与して、S3 リソースに対してアクションを実行できるようにします。S3 アクションの例は、バケット内のオブジェクトを読み取る `s3:GetObject` と、バケットにオブジェクトを書き込む `s3:PutObject` です。

条件キー

アクション以外に、条件が満たされた場合のみのアクセスを許可するように IAM 条件キーが制限されます。条件キーはオプションです。

> **Note**
>
> バケットポリシーなどのリソースベースのアクセスポリシー、またはアイデンティティベースのポリシーでは、以下を指定できます。
>
> - ポリシーステートメントの `Action` 要素内のアクションまたはアクションの配列。
> - ポリシーステートメントの `Effect` 要素で、`Allow` を指定してリストされたアクションを付与するか、`Deny` を指定してリストされたアクションをブロックできます。最小特権の使用をさらに維持するために、アクセスポリシーの `Effect` 要素の `Deny` のステートメントはできるだけ広義に、`Allow` ステートメントはできるだけ狭義にする必要があります。ポリシー条件ステートメントに含まれるアイデンティティに対して、オプトインに関するベストプラクティスを実装するための別の効果的な手段として、「`s3:*`」アクションとペアになった `Deny` 効果があります。
> - ポリシーステートメントの `Condition` 要素の条件キー。

アクセス管理のユースケース

Amazon S3 では、リソース所有者にアクセス権を付与するためのさまざまなツールを提供します。使用する S3 アクセス管理ツールは、共有する S3 リソース、アクセスを許可するアイデンティティ、許可または拒否するアクションによって異なります。S3 アクセス管理ツールの 1 つまたは組み合わせを使用して、S3 リソースへのアクセスを管理できます。

ほとんどの場合、アクセスポリシーを使用すると、アクセス許可を管理できます。アクセスポリシーは、バケットなどのリソース、または別の Amazon S3 リソース (S3 リソース) にアタッチされたリソースベースのポリシーに指定できます。アクセスポリシーは、アカウントの AWS Identity and

Access Management (IAM) ユーザー、グループ、ロールにアタッチされたアイデンティティベースのポリシーに指定することもできます。バケットポリシーがユースケースにより適している場合もあります。詳細については、「Amazon S3 のバケットポリシー」を参照してください。または、 AWS Identity and Access Management (IAM) を使用して、AWS アカウント 内に IAM ユーザー、グループ、ロールを作成し、アイデンティティベースのポリシーを介してバケットとオブジェクトへのアクセスを管理できます。詳細については、「Amazon S3 のアイデンティティベースのポリシー」を参照してください。

これらのアクセス管理オプションをナビゲートしやすくするために、Amazon S3 カスタマーの一般的なユースケースと各 S3 アクセス管理ツールの推奨事項を以下に示します。

AWS アカウント所有者が同じアカウント内のユーザーとのみバケットを共有したい

すべてのアクセス管理ツールで、この基本的なユースケースを満たすことができます。このユースケースでは、以下のアクセス管理ツールをお勧めします。

- バケットポリシー – 1 つのバケットまたは少数のバケットへのアクセスを許可する場合、またはバケットのアクセス許可がバケット間で類似している場合は、バケットポリシーを使用します。バケットポリシーでは、バケットごとに 1 つのポリシーを管理します。詳細については、「Amazon S3 のバケットポリシー」を参照してください。
- アイデンティティベースのポリシー – バケットごとにアクセス許可が異なるバケットが多数あり、管理するユーザーロールはわずかの場合は、ユーザー、グループ、またはロールに IAM ポリシーを使用できます。IAM ポリシーは、他の AWS リソースや Amazon S3 リソースへのユーザーアクセスを管理する場合にも適しています。詳細については、「例 1: バケット所有者がユーザーにバケットのアクセス許可を付与する」を参照してください。
- S3 Access Grants – S3 Access Grants を使用すると、S3 バケット、プレフィックス、またはオブジェクトへのアクセスを許可できます。S3 Access Grants では、さまざまなオブジェクトレベルのアクセス許可を大規模に指定できますが、バケットポリシーのサイズは 20 KB に制限されています。詳細については、「S3 Access Grants の開始方法」を参照してください。
- Access Points - バケットにアタッチされる名前付きのネットワークエンドポイントである Access Points を使用できます。バケットには最大 10,000 個のアクセスポイントをアタッチできます。各アクセスポイントでは、個別のアクセス許可やネットワークコントロールを適用することで、S3 オブジェクトへのアクセスをきめ細かく制御できます。詳細については、「Amazon S3 アクセスポイントを使用したデータアクセスの管理」を参照してください。

AWS アカウント所有者が別の AWS アカウントのユーザーとバケットまたはオブジェクトを共有したい (クロスアカウント)

別の AWS アカウントにアクセス許可を付与するには、バケットポリシーまたは次のいずれかの推奨アクセス管理ツールを使用する必要があります。このユースケースでは、アイデンティティベースのアクセスポリシーを使用できません。クロスアカウントアクセスの付与に関する詳細は、「Amazon S3 バケット内のオブジェクトへのクロスアカウントアクセスを提供するには、どうしたらいいですか?」を参照してください。

このユースケースでは、以下のアクセス管理ツールをお勧めします。

- バケットポリシー - バケットポリシーでは、バケットごとに 1 つのポリシーを管理します。詳細については、「Amazon S3 のバケットポリシー」を参照してください。
- S3 Access Grants – S3 Access Grants を使用すると、S3 バケット、プレフィックス、またはオブジェクトへのクロスアカウントアクセスを許可できます。S3 Access Grants を使用すると、さまざまなオブジェクトレベルのアクセス許可を大規模に指定できますが、バケットポリシーのサイズは 20 KB に制限されています。詳細については、「S3 Access Grants の開始方法」を参照してください。
- Access Points - バケットにアタッチされる名前付きのネットワークエンドポイントである Access Points を使用できます。バケットには最大 10,000 個のアクセスポイントをアタッチできます。各アクセスポイントでは、個別のアクセス許可やネットワークコントロールを適用することで、S3 オブジェクトへのアクセスをきめ細かく制御できます。詳細については、「Amazon S3 アクセスポイントを使用したデータアクセスの管理」を参照してください。

AWS アカウント所有者またはバケット所有者が、オブジェクトレベルまたはプレフィックスレベルでアクセス許可を付与する必要があり、これらのアクセス許可がオブジェクトごとまたはプレフィックスごとに異なる。

バケットポリシーでは、例えば、特定のキー名のプレフィックスを共有するバケット内のオブジェクト、または特定のタグを持つオブジェクトへのアクセスを許可できます。例えば、キー名プレフィックス `logs/` で始まるオブジェクトの読み取り権限を付与できます。ただし、アクセス許可がオブジェクトごとに異なる場合、特にバケットポリシーのサイズが 20 KB に制限されているため、バケットポリシーを使用して個々のオブジェクトへのアクセス許可を付与することは実用的でない可能性があります。

このユースケースでは、以下のアクセス管理ツールをお勧めします。

- S3 Access Grants – S3 Access Grants を使用すると、オブジェクトレベルまたはプレフィックスレベルのアクセス許可を管理できます。バケットポリシーとは異なり、S3 Access Grants を使用すると、さまざまなオブジェクトレベルのアクセス許可を大規模に指定できます。バケットポリシーのサイズは 20 KB に制限されています。詳細については、「S3 Access Grants の開始方法」を参照してください。
- Access Points – アクセスポイントを使用すると、オブジェクトレベルまたはプレフィックスレベルのアクセス許可を管理できます。Access Points は名前付きのネットワークエンドポイントで、バケットにアタッチされます。バケットには最大 10,000 個のアクセスポイントをアタッチできます。各アクセスポイントでは、個別のアクセス許可やネットワークコントロールを適用することで、S3 オブジェクトへのアクセスをきめ細かく制御できます。詳細については、「Amazon S3 アクセスポイントを使用したデータアクセスの管理」を参照してください。
- ACL – 特に ACL はオブジェクトごと負荷できる許可数が 100 個に制限されているため、アクセスコントロールリスト (ACL) の使用はお勧めしません。ただし、ACL を有効にした場合、バケット設定で [オブジェクト所有者] を [希望するバケット所有者] と [ACL 有効] に設定します。この設定では、新しいオブジェクトが `bucket-owner-full-control` 既定 ACL はオブジェクトライターではなく、バケット所有者によって自動的に所有されます。次に、XML 形式のアクセスポリシーであるオブジェクト ACL を使用して、他のユーザーにオブジェクトへのアクセスを許可できます。詳細については、「アクセスコントロールリスト (ACL) の概要」を参照してください。

AWS アカウント所有者またはバケット所有者が、特定のアカウント ID のみにバケットアクセスを制限したい

このユースケースでは、以下のアクセス管理ツールをお勧めします。

- バケットポリシー - バケットポリシーでは、バケットごとに 1 つのポリシーを管理します。詳細については、「Amazon S3 のバケットポリシー」を参照してください。
- Access Points – Access Points は名前付きのネットワークエンドポイントで、バケットにアタッチされます。バケットには最大 10,000 個のアクセスポイントをアタッチできます。各アクセスポイントでは、個別のアクセス許可やネットワークコントロールを適用することで、S3 オブジェクトへのアクセスをきめ細かく制御できます。詳細については、「Amazon S3 アクセスポイントを使用したデータアクセスの管理」を参照してください。

AWS アカウント 所有者またはバケット所有者が、データにアクセスするすべてのユーザーまたはアプリケーションに対して個別のエンドポイントを必要としている

このユースケースでは、以下のアクセス管理ツールをお勧めします。

- Access Points – Access Points は名前付きのネットワークエンドポイントで、バケットにアタッチされます。バケットには最大 10,000 個のアクセスポイントをアタッチできます。各アクセスポイントでは、個別のアクセス許可やネットワークコントロールを適用することで、S3 オブジェクトへのアクセスをきめ細かく制御できます。各アクセスポイントは、基になるバケットにアタッチされたバケットポリシーと連動して機能するカスタマイズされたアクセスポイントポリシーを適用します。詳細については、「Amazon S3 アクセスポイントを使用したデータアクセスの管理」を参照してください。

AWS アカウント所有者またはバケット所有者が、S3 の仮想プライベートクラウド (VPC)エンドポイントからのアクセスを管理する必要がある

Amazon S3 の仮想プライベートクラウド (VPC) エンドポイントは、S3 への接続のみを許可する VPC 内の論理エンティティです。このユースケースでは、以下のアクセス管理ツールをお勧めします。

- VPC 設定のバケット – バケットポリシーを使用して、バケットへのアクセスを許可されているユーザーと、バケットがアクセスできる VPC エンドポイントを制御できます。詳細については、「バケットポリシーを使用した VPC エンドポイントからのアクセスコントロール」を参照してください。
- アクセスポイント – アクセスポイントを設定する場合、アクセスポイントポリシーを使用できます。仮想プライベートクラウド (VPC) からのリクエストのみを受け付けるようにアクセスポイントを設定することで、プライベートネットワークへの Amazon S3 データアクセスを制限できます。また、アクセスポイントごとにカスタムのブロックパブリックアクセスを設定することもできます。詳細については、「Amazon S3 アクセスポイントを使用したデータアクセスの管理」を参照してください。

AWS アカウント所有者またはバケット所有者が静的ウェブサイトを公開する必要がある

S3 を使用すると、静的ウェブサイトをホストし、S3 バケットからホストされているウェブサイトのコンテンツを誰でも閲覧できるようになります。

このユースケースでは、以下のアクセス管理ツールをお勧めします。

- Amazon CloudFront – このソリューションでは、バケットのコンテンツへのすべてのパブリックアクセスを継続的にブロックしながら、Amazon S3 静的ウェブサイトをパブリックにホストできます。4 つすべての S3 の [パブリックアクセスをブロック] 設定を有効にしたまま、S3 静的ウェブサイトをホストする場合は、Amazon CloudFront のオリジンアクセスコントロール (OAC) を使用できます。Amazon CloudFront は、セキュアな静的ウェブサイトをセットアップするために必

要な機能を提供します。また、このソリューションを使用しない Amazon S3 静的ウェブサイトは、HTTP エンドポイントのみをサポートします。CloudFront は、耐久性に優れた Amazon S3 のあるストレージを使用し、HTTPS などの追加のセキュリティヘッダーを提供します。HTTPS では、通常の HTTP リクエストを暗号化し、一般的なサイバー攻撃から保護することで、セキュリティが強化されます。

詳細については、「Amazon CloudFront デベロッパーガイド」の「安全な静的ウェブサイトの使用開始」を参照してください。

- Amazon S3 バケットをパブリックにアクセス可能にする – パブリックにアクセス可能な静的ウェブサイトとして使用できるようにバケットを設定できます。

> **Warning**
>
> この方法はお勧めしていません。代わりに、Amazon CloudFront の一部として Amazon S3 静的ウェブサイトを使用することをお勧めします。詳細については、前のオプションを参照するか、「安全な静的ウェブサイトの使用開始」を参照してください。

Amazon CloudFront を使用せずに Amazon S3 静的ウェブサイトを作成するには、まずすべてのパブリックアクセスブロック設定をオフにする必要があります。静的ウェブサイトのバケットポリシーを記述するときは、`ListObject` または `PutObject` アクセス許可ではなく、`s3:GetObject` アクションのみを指定してください。これにより、ユーザーがバケット内のすべてのオブジェクトを表示したり、独自のコンテンツを追加したりできなくなります。詳細については、「ウェブサイトアクセスのアクセス許可の設定」を参照してください。

AWS アカウント所有者またはバケット所有者がバケットのコンテンツを公開したい

新しい Amazon S3 バケットを作成する場合、パブリックアクセスブロック設定はデフォルトで有効になっています。ブロックパブリックアクセスの詳細については、Amazon S3 ストレージへのパブリックアクセスのブロック を参照してください。

バケットへのすべてのパブリックアクセスを許可することはお勧めしません。ただし、特定のユースケースで設定する必要がある場合は、このユースケースで次のアクセス管理ツールを使用することをお勧めします。

- ブロックパブリックアクセス設定を無効にする – バケット所有者は、バケットへの認証されていないリクエストを許可できます。例えば、認証されていない Put Object リクエストは、バケットにパブリックバケットポリシーが適用されている場合や、バケットの ACL でパブリックアクセス

許可が付与されている場合は許可されます。認証されていないリクエストはすべて、他の任意の AWS ユーザー、または認証されていない匿名ユーザーによって行われます。このユーザーは、特定の正規ユーザー ID `65a011a29cdf8ec533ec3d1ccaae921c` によって、ACL で表されます。オブジェクトが `WRITE` または `FULL_CONTROL` にアップロードされた場合、特にすべてのユーザーグループまたは匿名ユーザーへのアクセスが許可されます。パブリックバケットポリシーとパブリックアクセスコントロールリスト (ACL) の詳細については、「パブリック」の意味 を参照してください。

AWS アカウント所有者またはバケット所有者がアクセスポリシーのサイズ制限を超えた

バケットポリシーとアイデンティティベースのポリシーには、どちらも 20 KB のサイズ制限があります。アクセス許可の要件が複雑な場合、このサイズ制限を超える可能性があります。

このユースケースでは、以下のアクセス管理ツールをお勧めします。

- Access Points — ユースケースで機能する場合、アクセスポイントを使用します。アクセスポイントでは、各バケットには複数の名前付きネットワークエンドポイントがあり、それぞれに基盤となるバケットポリシーで動作する独自のアクセスポイントポリシーがあります。ただし、アクセスポイントはバケットではなくオブジェクトに対してのみ動作し、クロスリージョンレプリケーションはサポートされていません。詳細については、「Amazon S3 アクセスポイントを使用したデータアクセスの管理」を参照してください。
- S3 Access Grants – S3 Access Grants を使用します。S3 Access Grants は、バケット、プレフィックス、またはオブジェクトへのアクセス権限を付与する非常に多数の許可をサポートしています。詳細については、「S3 Access Grants の開始方法」を参照してください。

AWS アカウント所有者または管理者ロールが、社内ディレクトリ内のユーザーまたはグループにバケット、プレフィックス、またはオブジェクトへの直接アクセスを許可したい

AWS Identity and Access Management (IAM) を介してユーザー、グループ、ロールを管理する代わりに、社内ディレクトリを AWS IAM Identity Center に追加できます。詳細については、「IAM Identity Center とは」を参照してください。

AWS IAM Identity Center に社内ディレクトリを追加したら、以下のアクセス管理ツールを使用して、社内ディレクトリアイデンティティに S3 リソースへのアクセス権を付与することをお勧めします。

- S3 Access Grants – S3 Access Grants を使用します。S3 Access Grants は、社内ディレクトリ内のユーザーまたはロールへのアクセスの付与をサポートしています。詳細については、「S3 Access Grants の開始方法」を参照してください。

AWS アカウント所有者またはバケット所有者が、CloudFront ログを S3 バケットに書き込むためのアクセス権をAWS CloudFront サービスに付与したい

このユースケースでは、以下のアクセス管理ツールをお勧めします。

- バケット ACL – バケット ACL の唯一の推奨されるユースケースは、 Amazon CloudFront `awslogsdelivery` アカウントなどの特定の AWS のサービスに許可を付与することです。ディストリビューションを作成または更新して、CloudFront ロギングを有効にすると、CloudFront はバケット ACL を更新して、バケットにログを書き込むための `awslogsdelivery` 許可を `FULL_CONTROL` アカウントに付与します。詳細については、「Amazon CloudFront デベロッパーガイド」の「標準ログ記録の設定とログファイルへのアクセスに必要なアクセス許可」を参照してください。ログを保存するバケットが S3 オブジェクト所有権のバケット所有者の強制設定を使用して ACL を無効にすると、CloudFront はバケットにログを書き込むことができません。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

バケット所有者が、他のユーザーによってバケットに追加されるオブジェクトを完全に制御したい。

バケットポリシー、アクセスポイント、または S3 Access Grants を使用して、バケットにオブジェクトをアップロードするためのアクセス権を他のアカウントに付与できます。バケットへのクロスアカウントアクセスが許可されている場合、バケットにアップロードされたオブジェクトに対して完全なコントロールを維持していることを確認できます。

このユースケースでは、以下のアクセス管理ツールをお勧めします。

- オブジェクト所有者 – バケットレベルの設定 [オブジェクト所有者] をデフォルトの [バケット所有者の強制] 設定のままにします。

アクセス管理のトラブルシューティング

以下のリソースは、S3 アクセス管理に関する問題のトラブルシューティングに役立ちます。

アクセス拒否 (403 Forbidden) エラーのトラブルシューティング

アクセス拒否の問題が発生した場合は、アカウントレベルとバケットレベルの設定を確認してください。また、アクセス権利を付与するために使用するアクセス管理機能をチェックして、ポリシー、

設定、または構成が正しいことを確認します。Amazon S3 のアクセス拒否 (403 禁止) エラーの一般的な原因の詳細については、「Amazon S3 でのアクセス拒否 (403 Forbidden) エラーのトラブルシューティング」を参照してください。

IAM Access Analyzer for S3

どのリソースも公開しない場合、またはリソースへのパブリックアクセスを制限する場合は、IAM Access Analyzer for S3 を使用します。Amazon S3 コンソールで、IAM Access Analyzer for S3 を使用して、パブリックアクセスまたは共有アクセスを許可するバケットアクセスコントロールリスト (ACL)、バケットポリシー、アクセスポイントポリシーを持つすべてのバケットを確認します。IAM Access Analyzer for S3 は、インターネットの任意のユーザーや他の AWS アカウント (組織外の AWS アカウント を含む) にアクセスを許可するように設定されているバケットに関して警告します。パブリックバケットまたは共有バケットごとに、パブリックアクセスや共有アクセスのソースとレベルを報告する結果が送信されます。

IAM Access Analyzer for S3 では、バケットへのすべてのパブリックアクセスを 1 つのアクションでブロックできます。特定のユースケースをサポートするためにパブリックアクセスが必要な場合を除き、バケットへのすべてのパブリックアクセスをブロックすることをお勧めします。すべてのパブリックアクセスをブロックする前に、アプリケーションがパブリックアクセスなしで正常に動作することを確認してください。詳細については、「Amazon S3 ストレージへのパブリックアクセスのブロック」を参照してください。

また、バケットレベルのアクセス許可の設定を参照して、詳細なアクセスレベルを設定することもできます。パブリックアクセスまたは共有アクセスを必要とする特定の検証済みユースケースについては、バケットの調査結果をアーカイブすることで、バケットをパブリックまたは共有とすることを確定して記録できます。これらのバケット設定はいつでも再確認および変更できます。結果は、監査目的で CSV レポートとしてダウンロードすることもできます。

IAM Access Analyzer for S3 は、Amazon S3 コンソールで追加料金なしで使用できます。IAM Access Analyzer for S3 は、AWS Identity and Access Management (IAM) IAM Access Analyzer を利用しています。Amazon S3 コンソールで IAM Access Analyzer for S3 を使用するには、IAM コンソールにアクセスして、IAM Access Analyzer で個別のリージョンごとにアカウントレベルのアナライザーを作成する必要があります。

IAM Access Analyzer for S3 の詳細については、IAM Access Analyzer for S3 を使用したバケットアクセスの確認 を参照してください。

ロギングとモニタリング

モニタリングは、アクセス障害を簡単にデバッグできるように、Amazon S3 ソリューションの信頼性、可用性、パフォーマンスを維持するうえで重要な部分です。ログ記録により、ユーザーが受け取ったエラーや、いつどのようなリクエストが行われたかを把握できます。AWS には、Amazon S3 リソースをモニタリングするための以下のようなツールがいくつかあります。

- AWS CloudTrail
- Amazon S3 アクセスログ
- AWS Trusted Advisor
- Amazon CloudWatch

詳細については、「Amazon S3 でのログ記録とモニタリング」を参照してください。

Amazon S3 用 Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御するために役立つ AWS のサービスです。IAM 管理者は、誰が認証 (サインイン) され、Amazon S3 リソースを使用する認可を受ける (許可がある) ことができるかを制御します。IAM は、追加費用なしで使用できる AWS のサービス です。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

トピック

- チュートリアル: ポリシーを使用した Amazon S3 リソースへのアクセスの管理
- Amazon S3 がリクエストを許可する仕組み
- Amazon S3 の AWS マネージドポリシー
- Amazon S3 ストレージレンズでのサービスにリンクされたロールの使用
- Amazon S3 アイデンティティとアクセスのトラブルシューティング

対象者

AWS Identity and Access Management (IAM) の使用方法は、Amazon S3 で行う作業に応じて異なります。

サービスユーザー – 業務を行うために Amazon S3 サービスを使用する場合、管理者が必要な認証情報および許可を提供します。業務のために使用する Amazon S3 機能が増えるにつれ、追加の許可が必要な場合があります。アクセスの管理方法を理解すると、管理者から適切な権限をリクエストするのに役に立ちます。Amazon S3 の機能にアクセスできない場合は、「Amazon S3 アイデンティティとアクセスのトラブルシューティング」を参照してください。

サービス管理者 – 社内の Amazon S3 リソースを担当している場合は、Amazon S3 に対する完全なアクセス権があると思われます。サービスのユーザーがどの Amazon S3 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社が Amazon S3 で IAM を使用する方法の詳細については、「Amazon S3 での IAM の機能」を参照してください。

IAM 管理者 – IAM 管理者には、Amazon S3 へのアクセスを管理するポリシーの作成方法の詳細を理解することが推奨されます。IAM で使用可能な Amazon S3 アイデンティティベースのポリシーの例を確認するには、「Amazon S3 のアイデンティティベースのポリシー」を参照してください。

アイデンティティによる認証

認証とは、アイデンティティ認証情報を使用して AWSにサインインする方法です。ユーザーは、AWS アカウントのルートユーザー、IAM ユーザーとして、または IAM ロールを引き受けることによって、認証される (AWSにサインインする) 必要があります。

ID ソースから提供された認証情報を使用して、フェデレーティッドアイデンティティとして AWS にサインインできます。AWS IAM Identity Centerフェデレーティッドアイデンティティの例としては、(IAM アイデンティティセンター) ユーザー、貴社のシングルサインオン認証、Google また

は Facebook の認証情報などがあります。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して AWSにアクセスする場合、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、AWS Management Consoleまたは AWS アクセスポータルにサインインできます。AWS へのサインインの詳細については、AWS サインインユーザーガイドの「AWS アカウントにサインインする方法」を参照してください。

プログラムを使用して AWS にアクセスする場合、AWSは Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) を提供し、認証情報を使用してリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに署名する推奨方法の使用については、「IAM ユーザーガイド」の「AWS API リクエストの署名」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、AWS は、アカウントのセキュリティを強化するために多要素認証 (MFA) を使用することをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「Multi-factor authentication」(多要素認証) および「IAM ユーザーガイド」の「AWSでの多要素認証 (MFA) の使用」を参照してください。

AWS アカウントのルートユーザー

AWS アカウントを作成する場合は、このアカウントのすべての AWS のサービスとリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。このアイデンティティは AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることによってアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、IAM ユーザーガイドのルートユーザー認証情報が必要なタスクを参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに対し、ID プロバイダーとのフェデレーションを使用して、一時的な認証情報の使用により、AWS のサービスにアクセスすることを要求します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダ、AWS Directory Service、Identity Center ディレクトリのユーザーか、または ID ソースから提供された認証

情報を使用して AWS のサービス にアクセスするユーザーです。フェデレーティッド ID が AWS アカウント にアクセスすると、ロールが継承され、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、AWS IAM Identity Centerを使用することをお勧めします。IAM アイデンティティセンター でユーザーとグループを作成するか、すべての AWS アカウント とアプリケーションで使用するために、独自の ID ソースで一連のユーザーとグループに接続して同期することもできます。IAM Identity Center の詳細については、「AWS IAM Identity Centerユーザーガイド」の「What is IAM Identity Center?」(IAM Identity Center とは) を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、1 人のユーザーまたは 1 つのアプリケーションに対して特定の許可を持つ AWS アカウント 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、IAM ユーザーガイドの長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションするを参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdminsという名前のグループを設定して、そのグループにIAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定の許可を持つ、AWS アカウント内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロールを切り替えることによって、AWS Management Consoleで IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、AWS CLIまたは AWS API オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「Creating a role for a third-party Identity Provider」(サードパーティーアイデンティティプロバイダー向けロールの作成) を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Centerユーザーガイド」の「アクセス許可セット」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。
- クロスサービスアクセス権 - 一部の AWS のサービスでは、他の AWS のサービスの機能を使用します。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービス またはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細に

ついては、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

- サービスにリンクされたロール - サービスにリンクされたロールは、AWS のサービスにリンクされたサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスにリンクされたロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション - EC2 インスタンスで実行され、AWS CLIまたは AWS API 要求を行っているアプリケーションの一時的な認証情報を管理するには、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付されたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、IAM ユーザーガイドのAmazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与するを参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、IAM ユーザーガイドの(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?を参照してください。

ポリシーを使用したアクセス権の管理

AWS でアクセスを制御するには、ポリシーを作成して AWS ID またはリソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けて、これらのアクセス許可を定義します。AWSは、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うと、これらのポリシーを評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWS に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、IAM ユーザーガイドのJSON ポリシー概要を参照してください。

管理者は AWSJSON ポリシーを使用して、だれが何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole`アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS Management Console、AWS CLI、または AWS API からロール情報を取得できます。

アイデンティティベースポリシー

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれます。マネージドポリシーは、AWS アカウント内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンポリシーです。マネージドポリシーには、AWS マネージドポリシーとカスタマー管理ポリシーがあります。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、IAM ユーザーガイドのマネージドポリシーとインラインポリシーの比較を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービス を含めることができます。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは IAM の AWSマネージドポリシーは使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、AWS WAF、および VPC は、ACL をサポートするサービスの例です。ACL の詳細については、Amazon Simple Storage Service デベロッパーガイドのアクセスコントロールリスト (ACL) の概要を参照してください。

その他のポリシータイプ

AWS では、他の一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、IAM ユーザーガイドのIAM エンティティのアクセス許可の境界を参照してください。
- サービスコントロールポリシー (SCP) - SCP は、AWS Organizationsで組織や組織単位 (OU) の最大許可を指定する JSON ポリシーです。AWS Organizationsは、顧客のビジネスが所有する複数の AWS アカウント をグループ化し、一元的に管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP はメンバーアカウントのエンティティに対する権限を制限します (各 AWS アカウントのルートユーザー など)。Organizations と SCP の詳細については、AWS Organizationsユーザーガイドの「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、IAM ユーザーガイドのセッションポリシーを参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、IAM ユーザーガイドのポリシーの評価ロジックを参照してください。

Amazon S3 での IAM の機能

IAM を使用して Amazon S3 へのアクセスを管理する前に、Amazon S3 で利用できる IAM 機能について理解しておく必要があります。

Amazon S3 で使用できる IAM の機能

IAM の機能	Amazon S3 のサポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	あり
ポリシーアクション	あり
ポリシーリソース	あり
ポリシー条件キー (サービス固有)	あり
ACL	あり
ABAC (ポリシー内のタグ)	部分的
一時的な認証情報	あり
転送アクセスセッション (FAS)	あり
サービスロール	あり
サービスリンクロール	部分的

Amazon S3 およびその他の AWS サービスがほとんどの IAM 機能との連携する方法の詳細については、「IAM ユーザーガイド」の「IAM と連携する AWS サービス」を参照してください。

Amazon S3 のアイデンティティベースのポリシー

アイデンティティベースポリシーをサポートする	あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素について学ぶには、IAM ユーザーガイドのIAM JSON ポリシーの要素のリファレンスを参照してください。

Amazon S3 のアイデンティティベースのポリシー例

Amazon S3 のアイデンティティベースのポリシー例を確認するには、「Amazon S3 のアイデンティティベースのポリシー」を参照してください。

Amazon S3 内のリソースベースのポリシー

リソースベースのポリシーのサポート	あり

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービス を含めることができます。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる AWS アカウント にある場合、信頼できるアカウントの IAM 管理者は、リソースにアクセスするための権限をプリンシパルエンティティ (ユーザーまたはロール) に付与する必要もあります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシー

をさらに付与する必要はありません。詳細については、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

Amazon S3 サービスは、バケットポリシー、アクセスポイントポリシー、およびアクセス許可をサポートします。

- バケットポリシーは、Amazon S3 バケットにアタッチされるリソースベースのポリシーです。リソースポリシーは、バケットに対してアクションを実行できるプリンシパルを定義します。
- アクセスポイントポリシーは、基になるバケットポリシーと組み合わせて評価されるリソースベースのポリシーです。
- アクセス許可は、Amazon S3 内のデータへのアクセス許可をプレフィックス、バケット、またはオブジェクトごとに定義するための簡略化されたモデルです。S3 Access Grants の詳細については、「S3 Access Grants でのアクセス管理」を参照してください。

バケットポリシーのプリンシパル

Principal エレメントは、リソースへのアクセスを許可または拒否するユーザー、アカウント、サービス、または他のエンティティを指定します。Principal を指定する例を以下に示します。詳細については、IAM ユーザーガイドのプリンシパルプリンシパル を参照してください。

AWS アカウントに許可を付与する

AWS アカウントに許可を付与するには、以下の形式を使用してアカウントを指定します。

```
"AWS":"account-ARN"
```

次に例を示します。

```
"Principal":{"AWS":"arn:aws:iam::AccountIDWithoutHyphens:root"}
```

```
"Principal":{"AWS":
["arn:aws:iam::AccountID1WithoutHyphens:root","arn:aws:iam::AccountID2WithoutHyphens:root"]}
```

IAM ユーザーにアクセス許可を付与する

アカウントの IAM ユーザーにアクセス許可を付与するには、"AWS":"*user-ARN*" の名前と値のペアを指定する必要があります。

```
"Principal":{"AWS":"arn:aws:iam::account-number-without-hyphens:user/username"}
```

ステップバイステップの手順を説明する詳細な例については、例 1: バケット所有者がユーザーにバケットのアクセス許可を付与する および 例 3: バケット所有者が自分の所有していないオブジェクトに対するアクセス許可を付与する を参照してください。

> **Note**
>
> バケットポリシーを更新した後に IAM ID を削除すると、バケットポリシーのプリンシパル要素には ARN の代わりに一意の ID が表示されます。これらの一意な ID は再利用されることがないため、すべてのポリシーステートメントから一意の ID を持つプリンシパルを安全に削除できます。一意の ID の詳細については、IAM ユーザーガイドの「IAM ID」を参照してください。

匿名アクセス許可を付与する

> **Warning**
>
> Simple Storage Service (Amazon S3) バケットへの匿名アクセスを付与するときは注意が必要です。匿名アクセスを付与すると、世界中のすべてのユーザーがバケットにアクセスできます。種類にかかわらず、S3 バケットへの匿名書き込みアクセスは一切付与しないことを強くお勧めします。

匿名アクセスと呼ばれるアクセス許可を全員に付与するには、`Principal` の値としてワイルドカード (`"*"`) を設定します。例えば、バケットを Web サイトとして設定する場合、以下のように、バケット内のすべてのオブジェクトを公開し、誰でもアクセスできるようにすることができます。

```
"Principal":"*"
```

```
"Principal":{"AWS":"*"}
```

リソースベースのポリシーで、`"Principal": "*"` 効果と共に `Allow` を使用すると、AWS にサインしていなくても、誰でもリソースにアクセスできるようになります。

リソースベースのポリシーで、`Allow` 効果と共に `"Principal" : { "AWS" : "*" }` を使用すると、同じパーティションのどのアカウントのルートユーザー、IAM ユーザー、引き受けたロールのセッション、フェデレーティッドユーザーでも、リソースにアクセスできるようになります。

匿名ユーザーの場合、これら 2 つの方法は同等です。詳細については、「IAM ユーザーガイド」の「すべてのプリンシパル」を参照してください。

ワイルドカードとして使用して、プリンシパルの名前または ARN の一部に一致させることはできません。

> **Important**
>
> 誰でも AWS アカウント を作成できるため、この 2 つの方法は機能は異なりますが、セキュリティレベルについては同等です。

リソースのアクセス許可の制限

リソースポリシーを使用して、それ以外の場合は IAM プリンシパルで利用できるリソースへのアクセスを制限することもできます。Deny ステートメントを使用してアクセスを防止します。

次の例では、安全な転送プロトコルが使用されていない場合はアクセスをブロックします。

```
{"Effect": "Deny",
 "Principal": "*",
 "Action": "s3:*",
 "Resource": <bucket ARN>,
 "Condition": {
   "Boolean": { "aws:SecureTransport" : "false"}
 }
}
```

このポリシーでは、この方法を使用して特定のアカウントやプリンシパルのみへのアクセスの拒否を試みるのではなく、"Principal": "*" を使用して、この制限をすべてのユーザーに適用することがベストプラクティスです。

CloudFront の URL を使用したアクセスの要求

Amazon S3 の URL の代わりに CloudFront の URL を使用することで、ユーザーが Amazon S3 のコンテンツのみにアクセスするように制限することができます。これを行うには、CloudFront オリジンアクセスコントロール (OAC) を作成します。その後、S3 データのアクセス許可を変更します。バケットポリシーでは、次のように CloudFront をプリンシパルとして設定できます。

```
"Principal":{"Service":"cloudfront.amazonaws.com"}
```

ポリシーの Condition 要素を使用して、S3 オリジンを含む CloudFront ディストリビューションに代わってリクエストが行われた場合にのみ CloudFront がバケットにアクセスできるようにします。

```
        "Condition": {
           "StringEquals": {
              "AWS:SourceArn":
"arn:aws:cloudfront::111122223333:distribution/CloudFront-distribution-ID"
           }
        }
```

CloudFront URL を使用した S3 アクセスの制限の詳細については、「Amazon CloudFront 開発者ガイド」の「Amazon Simple Storage Service オリジンへのアクセスの制限」を参照してください。Amazon CloudFront を使用する場合のセキュリティとプライバシーのメリットの詳細については、「コンテンツへのセキュアなアクセスとアクセス制限の設定」を参照してください。

Amazon S3 のリソースベースポリシーの例

- Amazon S3 バケットポリシーの例については、「Amazon S3 のバケットポリシー」を参照してください。
- アクセスポイントのポリシーの例については、「アクセスポイントを使用するための IAM ポリシーの設定」を参照してください。

Amazon S3 のポリシーアクション

ポリシーアクションに対するサポート	あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーのAction要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連する AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

以下に、S3 API オペレーションと必要なポリシーアクション間のさまざまなタイプのマッピング関係を示します。

- 同じ名前の 1 対 1 のマッピング。例えば、 PutBucketPolicy API オペレーションを使用するには、s3:PutBucketPolicy ポリシーアクションが必要です。
- 異なる名前の 1 対 1 のマッピング。例えば、 ListObjectsV2 API オペレーションを使用するには、s3:ListBucket ポリシーアクションが必要です。
- 1 対多のマッピング。例えば、HeadObject API オペレーションを使用するには、s3:GetObject が必要です。また、S3 オブジェクトロックを使用し、オブジェクトのリーガルホールドステータスまたは保持設定を取得したい場合は、HeadObject API オペレーションを使用する前に、対応する s3:GetObjectLegalHold または s3:GetObjectRetention ポリシーアクションも必要です。
- 1 対多のマッピング。例えば、ListObjectsV2 または HeadBucket API オペレーションを使用するには、s3:ListBucket ポリシーアクションが必要です。

ポリシーでの Amazon S3 のアクションの使用の一覧については、「サービス認証リファレンス」の「Amazon S3 で定義されるアクション」を参照してください。Amazon S3 API オペレーションの完全なリストについては、「Amazon Simple Storage Service API リファレンス」の「Amazon S3 API アクション」を参照してください。

Amazon S3 のポリシーアクションは、アクションの前に次のプレフィックスを使用します。

```
s3
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "s3:action1",
      "s3:action2"
         ]
```

バケットオペレーション

バケットオペレーションは、バケットリソースタイプで動作する S3 API オペレーションです。例: `CreateBucket`、`ListObjectsV2`、`PutBucketPolicy`。バケットオペレーションの S3 ポリシーアクションでは、バケットポリシーの `Resource` 要素または IAM アイデンティティベースのポリシーが、次の例形式の S3 バケットタイプの Amazon リソースネーム (ARN) 識別子である必要があります。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
```

次のバケットポリシーは、アカウント *12345678901* のユーザー *Akua* に ListObjectsV2 API オペレーションを実行し、S3 バケット内のオブジェクトを一覧表示する `s3:ListBucket` アクセス許可を付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to list objects in the bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
        }
    ]
}
```

アクセスポイントポリシーでのバケットオペレーション

アクセスポイントポリシーで付与されるアクセス許可は、基になるバケットで同じアクセス許可が許可される場合にのみ有効です。S3 アクセスポイントを使用する場合は、バケットからアクセスポイントにアクセスコントロールを委任するか、アクセスポイントポリシーで同じアクセス許可を基礎となるバケットのポリシーに追加する必要があります。詳細については、「アクセスポイントを使用するための IAM ポリシーの設定」を参照してください。アクセスポイントポリシーでは、バケットオペレーションの S3 ポリシーアクションで、次の形式の `Resource` 要素に `accesspoint` ARN を使用する必要があります。

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT"
```

次のアクセスポイントポリシーは、アカウント *12345678901* のユーザー *Akua* に、S3 アクセスポイント *DOC-EXAMPLE-ACCESS-POINT* を介して ListObjectsV2 API オペレーションを実行する s3:ListBucket アクセス許可を付与し、アクセスポイントに関連付けられたバケット内のオブジェクトを一覧表示します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to list objects in the bucket through access point",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-
ACCESS-POINT"
        }
    ]
}
```

> **Note**
>
> すべてのバケットオペレーションが S3 アクセスポイントでサポートされているわけではありません。詳細については、「S3 オペレーションとアクセスポイントの互換性」を参照してください。

オブジェクト操作

オブジェクトオペレーションは、オブジェクトリソースタイプに基づいて実行される S3 API オペレーションです。例: GetObject、PutObject、DeleteObject。オブジェクトオペレーションの S3 ポリシーアクションでは、ポリシーの Resource 要素を次の例の形式で S3 オブジェクト ARN にする必要があります。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
```

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/prefix/*"
```

> **Note**
>
> 前の例に示すように、オブジェクト ARN にはバケット名の後にスラッシュが含まれている必要があります。

次のバケットポリシーは、アカウント *12345678901* のユーザー *Akua* に PutObject API オペレーションを実行し、S3 バケット内のオブジェクトを一覧表示する s3:PutObject アクセス許可を付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to upload objects",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

アクセスポイントポリシーでのオブジェクトオペレーション

S3 アクセスポイントを使用してオブジェクトオペレーションへのアクセスを制御する場合、アクセスポイントポリシーを使用できます。アクセスポイントポリシーを使用するとき、オブジェクトオペレーションの S3 ポリシーアクションで、arn:aws:s3:region:account-id:accesspoint/access-point-name/object/resource 形式の Resource 要素に accesspoint ARN を使用する必要があります。アクセスポイントを使用するオブジェクトオペレーションの場合、アクセスポ

イント ARN 全体の後に /object/ 値を Resource 要素に含める必要があります。次に例を示します。

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT/
object/*"
```

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT/
object/prefix/*"
```

次のアクセスポイントポリシーは、アカウント *12345678901* のユーザー *Akua* に、アクセスポイント *DOC-EXAMPLE-ACCESS-POINT* を介してアクセスポイントに関連付けられたバケットのすべてのオブジェクトに対して GetObject API オペレーションを実行する s3:GetObject アクセス許可を付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to get objects through access point",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-
ACCESS-POINT/object/*"
        }
    ]
}
```

> **Note**
>
> すべてのオブジェクトオペレーションが S3 アクセスポイントでサポートされているわけではありません。詳細については、「S3 オペレーションとアクセスポイントの互換性」を参照してください。

アクセスポイントオペレーション

アクセスポイントオペレーションは、accesspoint リソースタイプで動作する S3 API オペレーションです。例: CreateAccessPoint、DeleteAccessPoint、GetAccessPointPolicy。アクセスポイントオペレーションの S3 ポリシーアクションは、バケットポリシーやアクセスポイントポリシーではなく、IAM アイデンティティベースのポリシーでのみ使用できます。アクセスポイントオペレーションでは、Resource 要素を次の例の形式の accesspoint ARN にする必要があります。

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT"
```

次の IAM アイデンティティベースのポリシーは、S3 アクセスポイント *DOC-EXAMPLE-ACCESS-POINT* で GetAccessPointPolicy API オペレーションを実行する s3:GetAccessPointPolicy アクセス許可を付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Grant permission to retrieve the access point policy of access
 point DOC-EXAMPLE-ACCESS-POINT",
            "Effect": "Allow",
            "Action": [
            "s3:GetAccessPointPolicy"
            ],
            "Resource": "arn:aws:s3:*:123456789012:access point/DOC-EXAMPLE-ACCESS-
POINT"
        }
    ]
}
```

アクセスポイントを使用してバケットオペレーションへのアクセスを制御する場合は、「アクセスポイントポリシーでのバケットオペレーション」を参照してください。オブジェクトオペレーションへのアクセスを制御するには、「アクセスポイントポリシーでのオブジェクトオペレーション」を参照してください。アクセスポイントポリシーの設定方法の詳細については、「アクセスポイントを使用するための IAM ポリシーの設定」を参照してください。

Object Lambda アクセスポイントオペレーション

Amazon S3 Object Lambda を使用すると、Amazon S3 GET、LIST、HEAD リクエストに独自のコードを追加して、データがアプリケーションに返されるときにそのデータを変更および処理できま

す。Object Lambda アクセスポイントを介したリクエストは、他のアクセスポイントを介したリクエストと同様に行うことができます。詳細については、「S3 Object Lambda を使用したオブジェクトの変換」を参照してください。

Object Lambda アクセスポイントオペレーションのポリシーを設定する方法についての詳細は、「Object Lambda アクセスポイントの IAM ポリシーの設定」を参照してください。

マルチリージョンアクセスポイントオペレーション

マルチリージョンアクセスポイントを使用すると、複数の AWS リージョン にある S3 バケットからのリクエストをアプリケーションが実行するために使用できるグローバルエンドポイントを作成できます。マルチリージョンアクセスポイントを使用して、単一のリージョンで使用するのと同じシンプルなアーキテクチャでマルチリージョンアプリケーションを構築し、世界中のどこでもこれらのアプリケーションを実行することができます。詳細については、「Amazon S3 マルチリージョンアクセスポイント」を参照してください。

マルチリージョンアクセスポイントオペレーションのポリシーを設定する方法の詳細については、「マルチリージョンアクセスポイントポリシーの例」を参照してください。

バッチジョブオペレーション

(バッチオペレーション) ジョブオペレーションは、ジョブリソースタイプで動作する S3 API オペレーションです。例えば、`DescribeJob` と `CreateJob` です。ジョブオペレーションの S3 ポリシーアクションは、バケットポリシーではなく、IAM アイデンティティベースのポリシーでのみ使用できます。また、ジョブオペレーションでは、IAM アイデンティティベースのポリシーの `Resource` 要素を次の例の形式の `job` ARN にする必要があります。

```
"Resource": "arn:aws:s3:*:123456789012:job/*"
```

次の IAM アイデンティティベースのポリシーは、S3 バッチオペレーションジョブ *DOC-EXAMPLE-JOB* に対して DescribeJob API オペレーションを実行する `s3:DescribeJob` アクセス許可を付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow describing the Batch operation job DOC-EXAMPLE-JOB",
            "Effect": "Allow",
```

```
            "Action": [
            "s3:DescribeJob"
            ],
            "Resource": "arn:aws:s3:*:123456789012:job/DOC-EXAMPLE-JOB"
        }
    ]
}
```

S3 Storage Lens 設定オペレーション

S3 Storage Lens 設定オペレーションの設定方法の詳細については、「Amazon S3 ストレージレンズアクセス許可」を参照してください。

アカウントオペレーション

アカウントオペレーションは、アカウントレベルで実行される S3 API オペレーションです。例えば、GetPublicAccessBlock (アカウント用) です。アカウントは、Amazon S3 で定義されるリソースタイプではありません。アカウントオペレーションの S3 ポリシーアクションは、バケットポリシーではなく、IAM アイデンティティベースのポリシーでのみ使用できます。また、アカウントオペレーションでは、IAM アイデンティティベースのポリシーの Resource 要素が "*" である必要があります。

次の IAM アイデンティティベースのポリシーは、アカウントレベルの GetPublicAccessBlock API オペレーションを実行し、アカウントレベルのパブリックアクセスブロック設定を取得する s3:GetAccountPublicAccessBlock アクセス許可を付与します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"Allow retrieving the account-level Public Access Block settings",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
       }
    ]
}
```

Amazon S3 のポリシーの例

- Amazon S3 のアイデンティティベースのポリシー例を確認するには、「Amazon S3 のアイデンティティベースのポリシー」を参照してください。
- Amazon S3 のリソースベースのポリシー例を確認するには、「Amazon S3 のバケットポリシー」および「アクセスポイントを使用するための IAM ポリシーの設定」を参照してください。

Amazon S3 のポリシーリソース

ポリシーリソースに対するサポート	あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

`Resource` JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、`Resource` または `NotResource` 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

一部の Amazon S3 API アクションは複数のリソースをサポートします。例えば、`s3:GetObject` は EXAMPLE-RESOURCE-1 と EXAMPLE-RESOURCE-2 にアクセスするため、プリンシパルには両方のリソースへのアクセス許可が必要です。複数リソースを単一ステートメントで指定するには、ARN をカンマで区切ります。

```
"Resource": [
      "EXAMPLE-RESOURCE-1",
      "EXAMPLE-RESOURCE-2"
```

Amazon S3 のリソースは、バケット、オブジェクト、アクセスポイント、またはジョブです。ポリシーでは、バケット、オブジェクト、アクセスポイント、またはジョブの Amazon リソースネーム (ARN) を使用してリソースを識別します。

Amazon S3 リソースタイプとその ARN の完全なリストについては、「サービス認証リファレンス」の「Amazon S3 で定義されるリソースタイプ」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Amazon S3 で定義されるアクション」を参照してください。

リソース ARN のワイルドカード

リソース ARN の一部にワイルドカードを使用することができます。ARN セグメント (コロンで区切られている部分) でワイルドカード文字 (* と ?) を使用できます。アスタリスク (*) は 0 個以上の文字の任意の組み合わせを表し、疑問符 (?) は任意の 1 文字を表します。各セグメントで複数の * または ? の文字を使用することができますが、ワイルドカードはセグメントをまたぐことができません。

- 以下の ARN は ARN の相対 ID の部分でワイルドカード * を使用して、examplebucket バケット内のすべてのオブジェクトを識別します。

```
arn:aws:s3:::examplebucket/*
```

- 次の ARN は * を使用して、S3 のすべてのバケットとオブジェクトを示しています。

```
arn:aws:s3:::*
```

- 以下の ARN は、relative-ID パートでワイルドカードと * および ? の両方を使用します。これにより、example1bucket、example2bucket、example3bucket など、バケット内のすべてのオブジェクトを識別します。

```
arn:aws:s3:::example?bucket/*
```

リソース ARN のポリシー変数

Amazon S3 の ARN では、ポリシー変数を使用することもできます。あらかじめ定義されているこれらの変数は、ポリシーの評価時に対応する値で置き換えられます。例えば、バケットをフォルダのコレクションとして構成し、ユーザーごとに別のフォルダを使用するとします。フォルダ名はユーザー名と同じです。各ユーザーに自分のフォルダに対するアクセス許可を付与するには、リソース ARN で以下のようにポリシー変数を指定します。

```
arn:aws:s3:::bucket_name/developers/${aws:username}/
```

実行時にポリシーが評価されると、リソース ARN の変数 ${aws:username} には、リクエストを行うユーザーのユーザー名が挿入されます。

Amazon S3 のポリシーの例

- Amazon S3 のアイデンティティベースのポリシー例を確認するには、「Amazon S3 のアイデンティティベースのポリシー」を参照してください。
- Amazon S3 のリソースベースのポリシー例を確認するには、「Amazon S3 のバケットポリシー」および「アクセスポイントを使用するための IAM ポリシーの設定」を参照してください。

Amazon S3 のポリシー条件キー

サービス固有のポリシー条件キーのサポート	あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成することで、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の Condition 要素を指定する場合、または 1 つの Condition 要素に複数のキーを指定する場合、AWSでは AND 論理演算子を使用してそれらを評価します。単一の条件キーに複数の値を指定する場合、AWS では OR 論理演算子を使用して条件を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、IAM ユーザーガイドのIAM ポリシーの要素: 変数およびタグを参照してください。

AWS はグローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、IAM ユーザーガイド の「AWS グローバル条件コンテキストキー」を参照してください。

各 Amazon S3 条件キーは、その条件を設定できる API でサポートされている同じ名前のリクエストヘッダーに対応します。Amazon S3 固有の条件キーでは、同じ名前のリクエストヘッダーの動作が指定されます。例えば、アクセス許可

`s3:GetObjectVersion`
に対して条件付きのアクセス許可を付与する条件キー `s3:VersionId` は、GET Object リクエストで設定する `versionId` クエリパラメータの動作を定義します。

Amazon S3 の条件キーのリストを確認するには、「サービス認証リファレンス」の「Amazon S3 の条件キー」を参照してください。どのアクションおよびリソースで条件キーを使用できるかについては、「Amazon S3 で定義されるアクション」を参照してください。

例: オブジェクトのアップロードを特定のストレージクラスのオブジェクトに制限する

アカウント ID が 123456789012 のアカウント A がバケットを所有しているとします。アカウント A の管理者は、アカウント A のユーザーである Dave に対して、STANDARD_IA ストレージクラスに保存されているバケットにのみオブジェクトのアップロードを許可するとします。オブジェクトのアップロードを特定のストレージクラスに制限するために、アカウント A の管理者は、次のバケットポリシーの例に示すように `s3:x-amz-storage-class` 条件キーを使用できます。

```
{
                "Version": "2012-10-17",
                "Statement": [
                  {
                    "Sid": "statement1",
                    "Effect": "Allow",
                    "Principal": {
                      "AWS": "arn:aws:iam::123456789012:user/Dave"
                    },
                    "Action": "s3:PutObject",
                    "Resource": "arn:aws:s3:::example-s3-bucket1/*",
                    "Condition": {
                      "StringEquals": {
                        "s3:x-amz-storage-class": [
                          "STANDARD_IA"
                        ]
                      }
                    }
                  }
                ]
            }
```

この例で、Condition ブロックは指定されたキーと値のペア `"s3:x-amz-acl":["public-read"]` に適用される `StringEquals` 条件を指定します。条件の表現に使用できる、事前に定義された一連のキーがあります。この例では、`s3:x-amz-acl` 条件キーを使用しています。この条件で

は、public-read の値が指定された x-amz-acl ヘッダーをすべての PUT Object リクエストに含めることがユーザーに求められます。

Amazon S3 のポリシーの例

- Amazon S3 のアイデンティティベースのポリシー例を確認するには、「Amazon S3 のアイデンティティベースのポリシー」を参照してください。
- Amazon S3 のリソースベースのポリシー例を確認するには、「Amazon S3 のバケットポリシー」および「アクセスポイントを使用するための IAM ポリシーの設定」を参照してください。

Amazon S3 での ACL

ACL のサポート	あり

Amazon S3 のアクセスコントロールリスト (ACL) は、どの AWS アカウント がリソースへのアクセス許可を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

> **Important**
>
> Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。

Amazon S3 での ACL を使用したアクセスコントロールについては、「ACL によるアクセス管理」を参照してください。

Amazon S3 での ABAC

ABAC (ポリシー内のタグ) のサポート	部分的

属性ベースのアクセス制御 (ABAC) は、属性に基づいてアクセス許可を定義する認可戦略です。AWS では、属性は タグ と呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール)、および多数の AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合にオペレーションを許可するように ABAC ポリシーをします。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、aws:ResourceTag/*key-name*、aws:RequestTag/*key-name*、または aws:TagKeys の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、IAM ユーザーガイドのABAC とは?を参照してください。ABAC をセットアップするステップを説明するチュートリアルについては、IAM ユーザーガイドの属性に基づくアクセスコントロール (ABAC) を使用するを参照してください。

タグに基づいて S3 バッチオペレーションジョブへのアクセスを制限するためのアイデンティティベースのポリシー例を確認するには、「ジョブタグを使用した S3 バッチ操作のアクセス許可の制御」を参照してください。

ABAC とオブジェクトタグ

ABAC ポリシーでは、オブジェクトは aws: タグの代わりに s3: タグを使用します。オブジェクトタグに基づいてオブジェクトへのアクセスをコントロールするには、以下のタグを使用してポリシーの条件要素でタグ情報を指定します。

- s3:ExistingObjectTag/*tag-key*
- s3:s3:RequestObjectTagKeys
- s3:RequestObjectTag/*tag-key*

アクセス許可ポリシーの例など、オブジェクトタグを使用したアクセスコントロールについては、「タグ付けとアクセスコントロールポリシー」を参照してください。

Amazon S3 での一時的な認証情報の使用

一時的な認証情報のサポート	あり

AWS のサービス には、一時的な認証情報を使用してサインインしても機能しないものがあります。一時的な認証情報を利用できる AWS のサービス を含めた詳細情報については、「IAM ユーザーガイド」の「IAM と連携する」AWS のサービスを参照してください。

ユーザー名とパスワード以外の方法で AWS Management Console にサインインする場合は、一時的な認証情報を使用していることになります。例えば、会社のシングルサインオン (SSO) リンクを使用して AWS にアクセスすると、そのプロセスは自動的に一時認証情報を作成します。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、IAM ユーザーガイドのロールへの切り替え (コンソール)を参照してください。

一時認証情報は、AWS CLIまたは AWS API を使用して手動で作成できます。作成後、一時認証情報を使用して AWS にアクセスできるようになります。AWSは、長期的なアクセスキーを使用する代わりに、一時認証情報を動的に生成することをお勧めします。詳細については、IAM の一時的セキュリティ認証情報を参照してください。

Amazon S3 の転送アクセスセッション

転送アクセスセッション (FAS) をサポート	あり

IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルとみなされます。一部のサービスを使用する際に、アクションを実行してから、別のサービスの別のアクションを開始することがあります。FAS は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービス またはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

- SSE-KMS を使用してオブジェクトを暗号化した場合、Amazon S3 は FAS を使用してオブジェクトを復号化するための呼び出しを AWS KMS に対して行います。詳細については、「AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用」を参照してください。
- S3 Access Grants も FAS を使用します。特定の ID の S3 データへのアクセス許可を作成すると、被付与者は S3 Access Grants に一時的な認証情報を要求します。S3 Access Grants は、AWS STS からリクエスタの一時的な認証情報を取得し、その認証情報をリクエスタに提供します。詳細については、「S3 Access Grants を介して Amazon S3 データへのアクセスをリクエストする」を参照してください。

Amazon S3 のサービスロール

サービスロールに対するサポート	あり

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

> **Warning**
>
> サービスロールのアクセス許可を変更すると、Amazon S3 が機能しなくなる可能性があります。サービスロールは、Amazon S3 で指示された場合のみ編集してください。

Amazon S3 のサービスリンクロール

サービスにリンクされたロールのサポート	部分的

サービスにリンクされたロールは、AWS のサービスにリンクされているサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスにリンクされたロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

Amazon S3 では、Amazon S3 Storage Lens 用のサービスにリンクされたロールをサポートしています。Amazon S3 でのサービスリンクロールの作成または管理の詳細については、「Amazon S3 ストレージレンズでのサービスにリンクされたロールの使用」を参照してください。

プリンシパルとしての Amazon S3 サービス

ポリシー内のサービス名	S3 機能	詳細情報
`s3.amazonaws.com`	S3 レプリケーション	ライブレプリケーションの設定

ポリシー内のサービス名	S3 機能	詳細情報
s3.amazonaws.com	S3 イベント通知	Amazon S3 イベント通知
s3.amazonaws.com	S3 インベントリ	Amazon S3 インベントリ
access-grants.s3.amazonaws.com	S3 Access Grants	ロケーションを登録する
batchoperations.s3.amazonaws.com	S3 バッチオペレーション	Amazon S3 バッチ操作に対するアクセス許可の付与
logging.s3.amazonaws.com	S3 サーバーアクセスのログ記録	Amazon S3 サーバーアクセスログを有効にします。
storage-lens.s3.amazonaws.com	S3 Storage Lens	データエクスポートで Amazon S3 Storage Lens のメトリクスを確認する

Amazon S3 のポリシーとアクセス許可

ここでは、Amazon S3 のバケットポリシーとユーザーポリシーの概要を示し、ポリシーの基本的なエレメントについて説明します。リストされた各エレメントは、その エレメントの詳細と使用方法の例にリンクしています。

Amazon S3 アクションの完全なリストについては、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

基本的に、ポリシーには以下のエレメントが含まれます。

- リソース — ポリシーが適用される Amazon S3 バケット、オブジェクト、アクセスポイント、またはジョブ。バケット、オブジェクト、アクセスポイント、またはジョブの Amazon リソースネーム (ARN) を使用してリソースを識別します。

  バケットレベルのオペレーションの例:

  - `"Resource": "arn:aws:s3:::bucket_name".`

  オブジェクトレベルのオペレーションの例:

- バケットのすべてのオブジェクトの "Resource": "arn:aws:s3:::*bucket_name/**"

- バケット内の特定のプレフィックスの下にあるオブジェクトの "Resource": "arn:aws:s3:::*bucket_name/prefix/**"

詳細については、「Amazon S3 のポリシーリソース」を参照してください。

- Action (アクション) – Amazon S3 では、各リソースに対して一連のオペレーションがサポートされています。許可 (または拒否) するリソースのオペレーションは、アクションキーワードを使用して識別します。

  例えば、s3:ListBucket のアクセス許可は、Amazon S3 のバケットの GET (ListObjects) オペレーションの使用を許可します。Amazon S3 アクションの使用に関する詳細については、「Amazon S3 のポリシーアクション」を参照してください。Amazon S3 のアクションの一覧については、Actions を参照してください。

- Effect (効力) – ユーザーが特定のアクションをリクエストした場合の効力です。許可または拒否のいずれかになります。

  リソースへのアクセスを明示的に許可していない場合、アクセスは暗黙的に拒否されます。リソースへのアクセスを明示的に拒否することもできます。別のポリシーでリソースへのアクセスが許可されているユーザーに対して、そのリソースへのアクセスを禁止できます。詳細については、IAM JSON ポリシーのエレメント: Effect を参照してください。

- プリンシパル – ステートメントのアクションやリソースへのアクセスが許可されているアカウントまたはユーザーを指します。バケットポリシーの場合、プリンシパルは、このアクセス許可の被付与者であるユーザー、アカウント、サービス、または他のエンティティです。詳細については、「バケットポリシーのプリンシパル」を参照してください。

- Condition (条件) – ポリシーが有効になる条件。Amazon S3 のアクセスポリシーの条件は、AWS 全体のキーと Amazon S3 固有のキーを使用して指定することができます。詳細については、「条件キーを使用したバケットポリシーの例」を参照してください。

次のバケットポリシーの例は、効果、プリンシパル、アクション、およびリソースエレメントを示しています。このポリシーは、*Account-ID* アカウントのユーザーの Akua に、s3:GetObject バケットに対する Amazon S3 の s3:GetBucketLocation、s3:ListBucket、awsexamplebucket1 のアクセス許可を付与します。

```
{
```

```
    "Version": "2012-10-17",
    "Id": "ExamplePolicy01",
    "Statement": [
        {
            "Sid": "ExampleStatement01",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Akua"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::awsexamplebucket1/*",
                "arn:aws:s3:::awsexamplebucket1"
            ]
        }
    ]
}
```

ポリシーの言語の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」および「IAM JSON ポリシーリファレンス」を参照してください。

アクセス許可の委任

AWS アカウントがリソースを所有している場合、そのアカウントは別の AWS アカウントにこれらの許可を付与できます。そのアカウントは、それらのアクセス許可またはそのサブセットを、アカウント内のユーザーに委任できます。これはアクセス許可の委任と呼ばれます。ただし、他のアカウントから許可を受け取るアカウントは、別の AWS アカウント にクロスアカウントで許可を委任することはできません。

Amazon S3 のバケットとオブジェクトの所有権

バケットとオブジェクトは Amazon S3 のリソースです。デフォルトでは、リソース所有者のみ、これらのリソースにアクセスできます。リソースの所有者とは、リソースを作成する AWS アカウントを指します。例:

- このようなリソースの所有者は、バケットの作成とオブジェクトのアップロードに使用する AWS アカウントです。

- AWS Identity and Access Management (IAM) ユーザーまたはロールの認証情報を使用してオブジェクトをアップロードする場合、オブジェクトの所有者は、そのユーザーやロールが属する AWS アカウントです。
- バケット所有者は、別の AWS アカウント (または別のアカウントのユーザー) に対して、オブジェクトをアップロードするためのクロスアカウントアクセス許可を付与できます。この場合、オブジェクトをアップロードする AWS アカウントが、それらのオブジェクトを所有します。バケット所有者には、他のアカウントが所有するオブジェクトに対するアクセス許可はありません。ただし、次のような例外があります。
  - バケット所有者が請求の支払いを行う場合、バケット所有者は、オブジェクトの所有者に関係なく、オブジェクトへのアクセスを拒否したり、バケット内のオブジェクトを削除したりすることができます。
  - バケット所有者は、オブジェクトの所有者に関係なく、オブジェクトをアーカイブしたり、アーカイブされたオブジェクトを復元したりすることができます。アーカイブはオブジェクトの格納に使用されるストレージクラスを指します。詳細については、「ストレージのライフサイクルの管理」を参照してください。

所有権とリクエスト認証

バケットに対するリクエストはすべて、認証済みまたは未認証です。認証済みのリクエストには、リクエストの送信元を認証する署名値を含める必要がありますが、未認証のリクエストには含みません。リクエストの認証の詳細については、リクエストの実行 を参照してください。

バケット所有者は認証されていないリクエストを許可できます。例えば、認証されていない PUT Object リクエストは、バケットにパブリックバケットポリシーが適用されている場合や、バケットの ACL で `All Users` グループまたは匿名ユーザーに `WRITE` または `FULL_CONTROL` のアクセス許可が付与されている場合に許可されます。パブリックバケットポリシーとパブリックアクセスコントロールリスト (ACL) の詳細については、「パブリック」の意味 を参照してください。

認証されていないリクエストはすべて、匿名ユーザーによって行われます。このユーザーは、特定の正規ユーザー ID `65a011a29cdf8ec533ec3d1ccaae921c` によって、ACL で表されます。認証されていないリクエストでオブジェクトがバケットにアップロードされている場合、所有者はその匿名ユーザーです。デフォルトのオブジェクト ACL では、`FULL_CONTROL` がオブジェクトの所有者として匿名ユーザーに付与されます。そのため、Amazon S3 では、認証されていないリクエストによるオブジェクトの取得やその ACL の変更を許可しています。

匿名ユーザーによってオブジェクトが変更されないように、匿名のパブリック書き込みをバケットに許可するバケットポリシーを実装したり、匿名ユーザーにバケットへの書き込みアクセスを許可した

りする ACL を使用しないことをお勧めします。この推奨事項を強制的に適用するには、Amazon S3 のパブリックアクセスのブロックを使用します。

ブロックパブリックアクセスの詳細については、Amazon S3 ストレージへのパブリックアクセスのブロック を参照してください。ACL の詳細については、アクセスコントロールリスト (ACL) の概要 を参照してください。

> **Important**
>
> 認証されたリクエストの実行に AWS アカウントのルートユーザー認証情報を使用しないことをお勧めします。代わりに、IAM ロールを作成し、このロールにフルアクセスを許可します。このロールを持つユーザーを管理者ユーザーと呼びます。AWS アカウントのルートユーザー認証情報ではなく、管理者ユーザーに割り当てられた認証情報を使用して AWS とやり取りし、バケットの作成、ユーザーの作成、アクセス許可の付与などのタスクを実行できます。詳細については、「IAM ユーザーガイド」の「AWS セキュリティ認証情報」および「IAM でのセキュリティのベストプラクティス」を参照してください。

Amazon S3 のバケットポリシー

バケットポリシーは、Amazon S3 バケットとその中のオブジェクトへのアクセス許可を付与できるリソースベースのポリシーです。バケット所有者のみが、ポリシーをバケットに関連付けることができます。バケットに添付された許可は、バケット所有者が所有するバケットのすべてのオブジェクトに適用されます。これらの許可は、他の AWS アカウント が所有するオブジェクトには適用されません。

S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、バケットにアップロードされたオブジェクトの所有権を制御し、アクセスコントロールリスト (ACL) を有効または無効にするのに使用できます。デフォルトでは、オブジェクト所有権は [バケット所有者の強制] により設定され、すべての ACL は無効になっています。バケット所有者は、バケット内のすべてのオブジェクトを所有し、ポリシーのみを使用してオブジェクトへのアクセスを管理します。

バケットポリシーは、JSON ベースの AWS Identity and Access Management (IAM) ポリシー言語を使用します。バケットポリシーを使用して、バケット内のオブジェクトに対する許可を追加または拒否できます。バケットポリシーは、ポリシーの要素に基づいて、リクエストを許可または拒否します。これらの要素には、リクエスタ、S3 アクション、リソース、およびリクエストの側面または条件 (リクエストの作成に使用された IP アドレスなど) が含まれます。

例えば、次のことを実行するバケットポリシーを作成できます。

- S3 バケットにオブジェクトをアップロードするためのクロスアカウントアクセス許可を付与するクロスアカウントアクセス許可を付与する
- バケット所有者であるユーザーが、アップロードされたオブジェクトを完全に管理できるようにする

詳細については、「Amazon S3 バケットポリシーの例」を参照してください。

> **Important**
>
> バケットポリシーを使用して、S3 ライフサイクルルールによる削除や移行を防ぐことはできません。例えば、バケットポリシーがすべてのプリンシパルのすべてのアクションを拒否する場合でも、S3 ライフサイクル設定は通常どおり機能します。

このセクションのトピックでは、具体例と、S3 コンソールにバケットポリシーを追加する方法について説明します。アイデンティティベースのポリシーについては、「Amazon S3 のアイデンティティベースのポリシー」を参照してください。バケットポリシー言語の詳細については、「Amazon S3 のポリシーとアクセス許可」を参照してください。

トピック

- Amazon S3 コンソールを使用したバケットポリシーの追加
- バケットポリシーを使用した VPC エンドポイントからのアクセスコントロール
- Amazon S3 バケットポリシーの例
- 条件キーを使用したバケットポリシーの例

Amazon S3 コンソールを使用したバケットポリシーの追加

AWS Policy Generator および Amazon S3 コンソールを使用して、新しいバケットポリシーを追加したり、既存のバケットポリシーを編集したりできます。バケットポリシーはリソースベースの AWS Identity and Access Management (IAM) ポリシーです。バケットポリシーをバケットに追加して、バケットおよびバケット内のオブジェクトに対するアクセス許可を、他の AWS アカウントまたは IAM ユーザーに付与できます。オブジェクトのアクセス許可は、バケット所有者が作成したオブジェクトにのみ適用されます。バケットポリシーの詳細については、Amazon S3 用 Identity and Access Management を参照してください。

ポリシーを保存する前に、AWS Identity and Access Management Access Analyzer でセキュリティ警告、エラー、一般的な警告、および提案を解決してください。IAM Access Analyzer は、IAM ポリシーの文法とベストプラクティスに照らしてポリシーチェックを行います。これらのチェックにより、機能的でセキュリティのベストプラクティスに準拠したポリシーを作成するのに、役立つ結果と実行可能なレコメンデーションが示されます。IAM Access Analyzer を使用したポリシーの検証の詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer のポリシーの検証」を参照してください。IAM Access Analyzer によって返される警告、エラー、および提案のリストを表示するには、「IAM Access Analyzer ポリシーチェックリファレンス」を参照してください。

ポリシーのエラーのトラブルシューティングに関するガイダンスについては、「Amazon S3 でのアクセス拒否 (403 Forbidden) エラーのトラブルシューティング」を参照してください。

バケットポリシーを作成または編集するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [Buckets (バケット)] リストで、バケットポリシーを作成するバケットの名前、またはバケットポリシーを編集するバケットの名前を選択します。
4. [アクセス許可] タブを選択します。
5. [バケットポリシー] で [編集] を選択します。[Edit bucket policy] (バケットポリシーを編集) ページが表示されます。
6. [Edit bucket policy] (バケットポリシーを編集) ページで、次の操作を実行します。

   - 「Amazon S3 ユーザーガイド」でバケットポリシーの例を確認するには、[Policy examples] (ポリシーの例) を選択してください。
   - ポリシーを自動的に生成するか、Policy] (ポリシー) セクションで JSON を編集するには、[Policy Generator] を選択します。

   [Policy Generator] を選択すると、AWS Policy Generator が新しいウィンドウで開きます。

   a. [Select Type of Policy] (ポリシーの種類の選択) の [AWS Policy Generator] ページで、[S3 Bucket Policy] (S3 バケットポリシー)を選択します。
   b. 提供されたフィールドに情報を入力してステートメントを追加し、ステートメントの追加を選択します。このステップを、追加するステートメントの数だけ繰り返します。ポリシーステートメントの詳細については、IAM ユーザーガイドの IAM JSON ポリシーのエレメントのリファレンスを参照してください。

> **Note**
>
> わかりやすいように、[バケットポリシーの編集] ページでは、現在のバケットの [バケット ARN] (Amazon リソースネーム) が [ポリシー] テキストフィールドの上に表示されます。この ARN をコピーして、AWSポリシージェネレータのステートメントで使用できます。

c. ステートメントの追加が完了したら、ポリシーの生成を選択します。

d. 生成されたポリシーテキストをコピーし、[閉じる] を選択すると、Amazon S3 コンソールのバケットポリシーの編集ページに戻ります。

7. [Policy] (ポリシー) ボックスで、既存のポリシーを編集するか、AWS Policy Generator からバケットポリシーを貼り付けます。ポリシーを保存する前に、セキュリティ警告、エラー、一般的な警告、および提案を解決してください。

> **Note**
>
> バケットポリシーのサイズは 20 KB に制限されています。

8. (オプション) 新しいポリシーがリソースへのパブリックアクセスおよびクロスアカウントアクセスにどのように影響するかをプレビューするには、[Preview external access] (外部アクセスをプレビュー) を選択します。ポリシーを保存する前に、新しい IAM Access Analyzer の結果が導入されているかどうかや、既存の結果を解決するかどうかを確認できます。アクティブなアナライザーが表示されない場合は、[Go to Access Analyzer] (Access Analyzer に移動) を選択し、[IAM Access Analyzer] でアカウントアナライザーを作成します。詳細については、IAM ユーザーガイドのアクセスのプレビューを参照してください。

9. [Save changes] (変更の保存) を選択すると、[Permissions] (アクセス許可) タブへ戻ります。

バケットポリシーを使用した VPC エンドポイントからのアクセスコントロール

Amazon S3 のバケットポリシーを使用して、特定の仮想プライベートクラウド (VPC) エンドポイントまたは特定の VPC からのバケットへのアクセスを管理できます。このセクションでは、VPC エンドポイントからの Amazon S3 のバケットへのアクセスを管理するために使用できるバケットポリシーの例を示します。VPC エンドポイントの設定方法については、VPC ユーザーガイドの VPC エンドポイントを参照してください。

VPC を使用すると、定義した仮想ネットワーク内で AWS のリソースを起動できます。VPC エンドポイントにより、VPC と他の AWS のサービス 間のプライベート接続を作成できます。このプライベート接続では、インターネットアクセス、仮想プライベートネットワーク (VPN) 接続、NAT インスタンス、または AWS Direct Connect を介した接続は必要ありません。

Amazon S3 の VPC エンドポイントは、Amazon S3 への接続のみを許可する VPC 内の論理エンティティです。VPC エンドポイントはリクエストを Amazon S3 にルーティングし、レスポンスを VPC にルーティングします。VPC エンドポイントはリクエストのルーティング方法のみを変更します。Amazon S3 のパブリックエンドポイントと DNS 名は、VPC エンドポイントでも使用できます。Amazon S3 での VPC エンドポイントの使用の詳細については、「VPC ユーザーガイド」の「Gateway endpoints」および「Gateway endpoints for Amazon S3」を参照してください。

Amazon S3 の VPC エンドポイントは、Amazon S3 のデータへのアクセスを管理する 2 つの方法を提供します。

- 特定の VPC エンドポイントを通じて許可されるリクエスト、ユーザー、またはグループを管理できます。このタイプのアクセス管理については、「VPC ユーザーガイド」の「Controlling access to VPC endpoints using endpoint policies」を参照してください。
- Amazon S3 のバケットポリシーを使用して、バケットへのアクセスを許可する VPC または VPC エンドポイントを管理できます。この種類のバケットポリシーのアクセスコントロールの例については、アクセス制限に関する次のトピックを参照してください。

トピック

- 特定の VPC エンドポイントへのアクセスの制限
- 特定の VPC へのアクセスの制限

> **Important**
>
> このセクションで説明している VPC エンドポイントに Amazon S3 のバケットポリシーを適用すると、バケットへのアクセスが意図せずにブロックされる場合があります。特に VPC エンドポイントからの接続に対するバケットのアクセスを制限することを目的としたバケットのアクセス許可により、バケットへのすべての接続がブロックされる場合があります。この問題を修正する方法については、「AWS Support 情報センター」の「バケットポリシーの VPC または VPC エンドポイント ID が間違っています。ポリシーを修正してバケットに再度アクセスできるようにするにはどうすればよいですか?」を参照してください。

特定の VPC エンドポイントへのアクセスの制限

以下に、awsexamplebucket1 という特定のバケットに対するアクセスを ID が vpce-1a2b3c4d の VPC エンドポイントからのみに制限する Amazon S3 のバケットポリシーの例を示します。指定されたエンドポイントが使用されていない場合、ポリシーはバケットに対するすべてのアクセスを拒否します。aws:SourceVpce 条件はエンドポイントを指定します。aws:SourceVpce 条件では、VPC エンドポイントの ID のみが必要で、VPC エンドポイントのリソースの Amazon リソースネーム (ARN) は必要ありません。ポリシーで条件を使用する方法の詳細については、条件キーを使用したバケットポリシーの例 を参照してください。

> **Important**
>
> - 次のポリシーの例を使用する前に、VPC エンドポイントの ID をユースケースに応じた値に置き換えてください。そうしないと、バケットにアクセスできません。
> - このポリシーは、コンソールリクエストが指定の VPC エンドポイントを経由していない場合、指定先のバケットへのコンソールアクセスを無効にします。

```
{
   "Version": "2012-10-17",
   "Id": "Policy1415115909152",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPCE-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::awsexamplebucket1",
                    "arn:aws:s3:::awsexamplebucket1/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:SourceVpce": "vpce-1a2b3c4d"
         }
       }
     }
   ]
}
```

特定の VPC へのアクセスの制限

aws:SourceVpc 条件を使用して、特定の VPC へのアクセスを制限するバケットポリシーを作成できます。これは、同じ VPC で複数の VPC エンドポイントを設定していて、すべてのエンドポイントで Amazon S3 のバケットへのアクセスを管理したい場合に便利です。以下は、VPC vpc-111bbb22 の外側から awsexamplebucket1 およびそのオブジェクトへのアクセスを拒否するポリシーの例です。指定された VPC が使用されていない場合、ポリシーはバケットに対するすべてのアクセスを拒否します。このステートメントは、バケットへのアクセス許可を付与しません。アクセス許可を付与するには、別の Allow ステートメントを追加する必要があります。vpc-111bbb22 条件キーでは、VPC リソースの ARN は不要で、VPC ID のみが必要です。

> **Important**
>
> - 次のポリシーの例を使用する前に、VPC の ID をユースケースに応じた値に置き換えてください。そうしないと、バケットにアクセスできません。
> - このポリシーは、コンソールリクエストが指定の VPC を経由していない場合、指定先のバケットへのコンソールアクセスを無効にします。

```
{
   "Version": "2012-10-17",
   "Id": "Policy1415115909153",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPC-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::awsexamplebucket1",
                    "arn:aws:s3:::awsexamplebucket1/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:SourceVpc": "vpc-111bbb22"
         }
       }
     }
   ]
}
```

Amazon S3 バケットポリシーの例

Amazon S3 バケットポリシーを使用すると、バケット内のオブジェクトへのアクセスを保護して、適切な権限を持つユーザーだけがアクセスできるようにすることができます。適切な権限を持たない認証済みユーザーが Amazon S3 リソースにアクセスできないようにすることもできます。

このセクションでは、バケットポリシーの一般的なユースケース例を紹介します。これらのサンプルポリシーは、*example-s3-bucket* をリソース値として使用します。これらのポリシーをテストするには、*user input placeholders* をお客様の情報 (バケット名など) と置き換えます。

オブジェクトのセットに対するアクセス許可を付与または拒否するために、Amazon リソースネーム (ARN) やその他の値でワイルドカード文字 (*) を使用できます。例えば、共通のプレフィックスで始まるか、.html などの特定の拡張子で終わるオブジェクトのグループへのアクセスをコントロールできます。

AWS Identity and Access Management (IAM) ポリシー言語については、「Amazon S3 のポリシーとアクセス許可」を参照してください。

> **Note**
>
> Amazon S3 コンソールを使用して許可をテストするときには、コンソールに必要な s3:ListAllMyBuckets、s3:GetBucketLocation、s3:ListBucket を追加で付与する必要があります。コンソールを使用してユーザーにアクセス許可を付与してテストする例の解説については、「ユーザーポリシーを使用したバケットへのアクセスの制御」を参照してください。

バケットポリシーを作成するためのその他のリソースには、以下が含まれます。

- バケットポリシーの作成時に使用できる IAM ポリシーアクション、リソース、および条件キーの完全なリストについては、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。
- S3 ポリシーの作成に関するガイダンスについては、「Amazon S3 コンソールを使用したバケットポリシーの追加」を参照してください。
- ポリシーのエラーのトラブルシューティングを行うには、「Amazon S3 でのアクセス拒否 (403 Forbidden) エラーのトラブルシューティング」を参照してください。

トピック

- 任意の匿名ユーザーへの読み取り専用アクセス許可の付与
- 暗号化が必要
- 定型の ACL を使ったバケットの管理
- オブジェクトタグ付けによるオブジェクトアクセスの管理
- グローバル条件キーによるオブジェクトアクセスの管理
- 特定の IP アドレスに基づくアクセス管理
- HTTP または HTTPS リクエストに基づくアクセス管理
- 特定のフォルダへのユーザーアクセスの管理
- アクセスログへのアクセス管理
- Amazon CloudFront の OAI へのアクセス管理
- Amazon S3 ストレージレンズへのアクセス管理
- S3 インベントリ、S3 分析、および S3 インベントリレポートの権限管理
- MFA が必要
- ユーザーによるオブジェクトの削除を禁止する

任意の匿名ユーザーへの読み取り専用アクセス許可の付与

ポリシー設定を使用して、任意の匿名ユーザーにアクセス権を付与できます。これは、バケットを静的ウェブサイトとして設定する場合に便利です。そのためには、バケットへのパブリックアクセスのブロックを無効にする必要があります。これを行う方法と必要なポリシーの詳細については、「ウェブサイトアクセスのアクセス許可の設定」を参照してください。同じ目的でより制限の厳しいポリシーを設定する方法については、「AWS 情報センター」の「Amazon S3 バケットの一部のオブジェクトにパブリック読み取りアクセス権を付与するにはどうすればよいですか?」を参照してください。

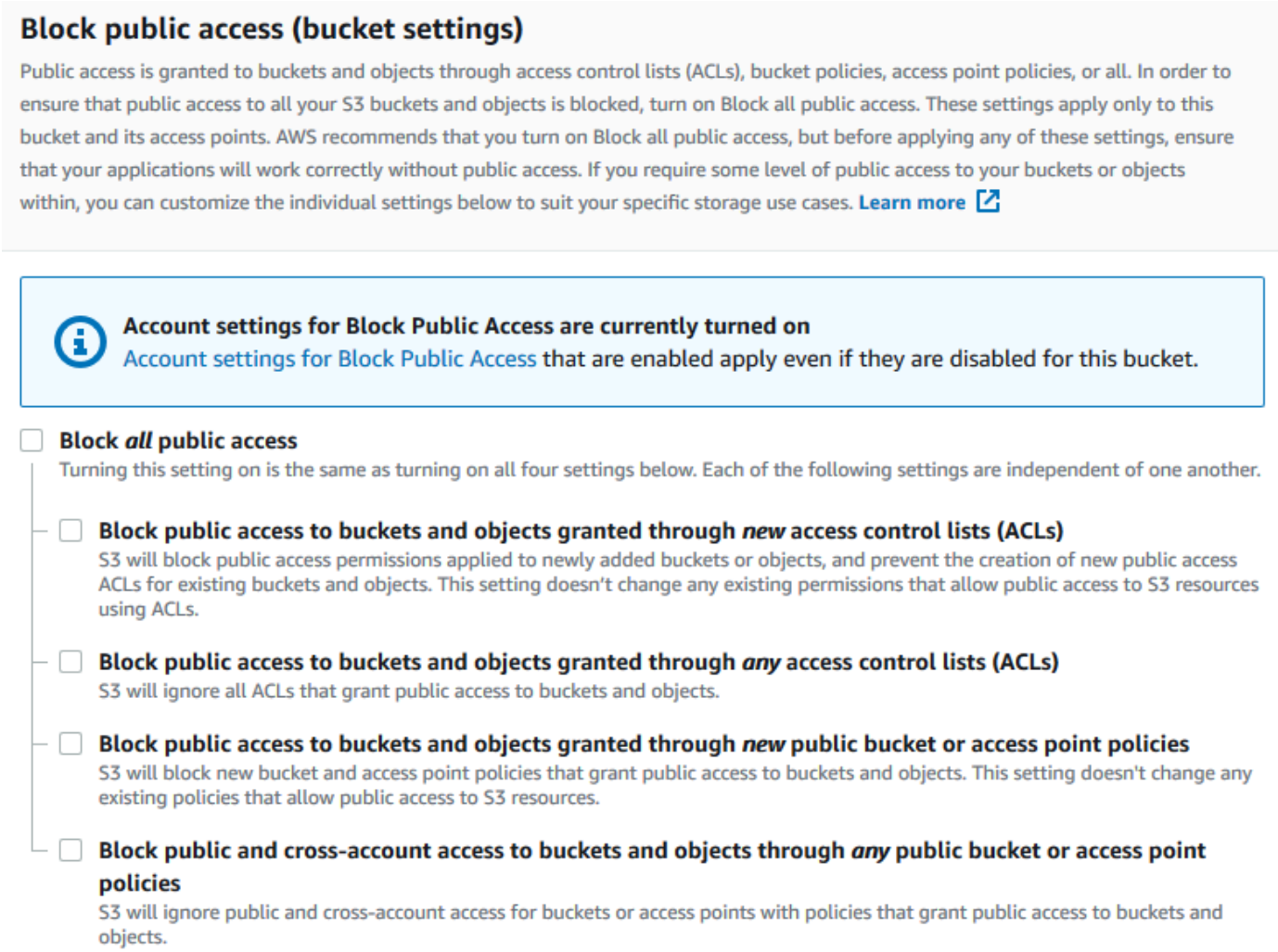
デフォルトでは、Amazon S3 はアカウントとバケットへのパブリックアクセスをブロックします。バケットを使用して静的ウェブサイトをホストする場合は、以下のステップを使用して、パブリックアクセスブロック設定を編集できます。

> **Warning**
>
> このステップを完了する前に「Amazon S3 ストレージへのパブリックアクセスのブロック」を読んで、パブリックアクセスを許可することに伴うリスクを理解し、了承します。パブリックアクセスブロック設定をオフにしてバケットをパブリックにすると、インターネット

上のだれでもバケットにアクセスできるようになります。バケットへのすべてのパブリックアクセスをブロックすることをお勧めします。

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 静的ウェブサイトとして設定されたバケットの名前を選択します。
3. [Permissions (アクセス許可)] を選択します。
4. [ブロックパブリックアクセス (バケット設定)] で [編集] を選択します。
5. [Block all public access (すべてのパブリックアクセスをブロックする)] をクリアし、[Save changes (変更の保存)] を選択します。

> **Warning**
>
> このステップを完了する前に、「Amazon S3 ストレージへのパブリックアクセスのブロック」を確認し、パブリックアクセスの許可に伴うリスクを理解したうえで了承してください。パブリックアクセスブロック設定をオフにしてバケットをパブリックにすると、インターネット上のだれでもバケットにアクセスできるようになります。バケットへのすべてのパブリックアクセスをブロックすることをお勧めします。



Amazon S3 は、バケットのパブリックアクセスブロック設定をオフにします。パブリックで静的ウェブサイトを作成するには、バケットポリシーを追加する前に、アカウントのブロックパブリックアクセス設定を編集する必要があります。パブリックアクセスのブロックのアカウント設定が現在有効になっている場合は、[Block public access (bucket settings) (パブリックアクセスのブロック (バケット設定))] の下にメモが表示されます。

暗号化が必要

バケットに書き込まれるすべてのオブジェクトに SSE-KMS が必要

次のポリシー例では、バケットに書き込まれるすべてのオブジェクトを、AWS Key Management Service (AWS KMS) キーを使用したサーバー側の暗号化 (SSE-KMS) を使用して暗号化することを

要求しています。オブジェクトが SSE-KMS で暗号化されていない場合、リクエストは拒否されます。

```
{
"Version": "2012-10-17",
"Id": "PutObjPolicy",
"Statement": [{
  "Sid": "DenyObjectsThatAreNotSSEKMS",
  "Principal": "*",
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
  "Condition": {
    "Null": {
      "s3:x-amz-server-side-encryption-aws-kms-key-id": "true"
    }
  }
}]
}
```

バケットに書き込まれるすべてのオブジェクトに特定の AWS KMS key を使用する SSE-KMS が必要

次のポリシー例では、特定の KMS キー ID を使用して SSE-KMS で暗号化されていないオブジェクトはバケットに書き込まれません。オブジェクトがリクエストごとのヘッダーまたはバケットのデフォルト暗号化を使用して SSE-KMS で暗号化されている場合でも、特定の KMS キーで暗号化されていないオブジェクトはバケットに書き込めません。この例で使用している KMS キー ARN を必ずお客様の KMS キー ARN に置き換えてください。

```
{
"Version": "2012-10-17",
"Id": "PutObjPolicy",
"Statement": [{
  "Sid": "DenyObjectsThatAreNotSSEKMSWithSpecificKey",
  "Principal": "*",
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
  "Condition": {
    "ArnNotEqualsIfExists": {
      "s3:x-amz-server-side-encryption-aws-kms-key-id": "arn:aws:kms:us-
east-2:111122223333:key/01234567-89ab-cdef-0123-456789abcdef"
```

```
    }
  }
}]
}
```

定型の ACL を使ったバケットの管理

オブジェクトをアップロードしたり、パブリックアクセス用のオブジェクト ACL を設定したりする権限を複数のアカウントに付与する

以下のユーザーポリシーの例は、複数の AWS アカウント に `s3:PutObject` アクセス許可および `s3:PutObjectAcl` アクセス許可を付与します。また、このポリシーの例では、これらのオペレーションのリクエストに `public-read` という既定のアクセスコントロールリスト (ACL) が含まれることを要求します。詳細については、Amazon S3 のポリシーアクションおよびAmazon S3 のポリシー条件キーを参照してください。

> **Warning**
>
> `public-read` という既定の ACL を使用すると、バケット内のオブジェクトを世界中の誰でも見ることができます。Amazon S3 のバケットへの匿名アクセスを許可したり、パブリックアクセスブロック設定を無効にしたりする場合は注意が必要です。匿名アクセスを付与すると、世界中のすべてのユーザーがバケットにアクセスできます。静的なウェブサイトのホスティングなどで特に必要な場合を除いて、Amazon S3 のバケットへの匿名アクセスは許可しないことをお勧めします。静的ウェブサイトホスティングのパブリックアクセスをブロックする設定を有効にする場合は、「チュートリアル: Amazon S3 での静的ウェブサイトの設定」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AddPublicReadCannedAcl",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root",
                    "arn:aws:iam::444455556666:root"
                ]
            },
```

```
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": [
                        "public-read"
                    ]
                }
            }
        }
    ]
}
```

バケット所有者はフルコントロール権限を持ちながら、オブジェクトをアップロードするためのクロスアカウントアクセス許可を付与する

以下の例は、アップロードされたオブジェクトを完全に制御しながら、別の AWS アカウント がバケットにオブジェクトをアップロードできるようにする方法を示しています。このポリシーでは、特定の AWS アカウント (*111122223333*) に、アップロード時に bucket-owner-full-control の既定 ACL が含まれている場合にのみ、オブジェクトをアップロードする機能が付与されます。ポリシーの StringEquals 条件は、要件を表現する s3:x-amz-acl 条件キーを指定します。詳細については、「Amazon S3 のポリシー条件キー」を参照してください。

```
{
   "Version":"2012-10-17",
   "Statement":[
     {
       "Sid":"PolicyForAllowUploadWithACL",
       "Effect":"Allow",
       "Principal":{"AWS":"111122223333"},
       "Action":"s3:PutObject",
       "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
       "Condition": {
         "StringEquals": {"s3:x-amz-acl":"bucket-owner-full-control"}
       }
     }
   ]
}
```

オブジェクトタグ付けによるオブジェクトアクセスの管理

特定のタグキーと値を持つオブジェクトの読み取り権限のみをユーザーに許可する

以下のアクセス許可ポリシーでは、environment: production タグキーと値を持つオブジェクトのみを読み取れるように制限しています。このポリシーは s3:ExistingObjectTag 条件キーを使用してタグキーと値を指定します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Principal":{
            "AWS":"arn:aws:iam::111122223333:role/JohnDoe"
         },
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
         "Condition":{
            "StringEquals":{
               "s3:ExistingObjectTag/environment":"production"
            }
         }
      }
   ]
}
```

ユーザーが追加できるオブジェクトタグキーを制限する

次のポリシーの例では、s3:PutObjectTagging アクションを実行するアクセス許可をユーザーに付与します (ユーザーが既存のオブジェクトにタグを追加することができます)。この条件は s3:RequestObjectTagKeys 条件キーを使用して、Owner や CreationDate などの許可されたタグキーを指定します。詳細については、「IAM ユーザーガイド」の「複数のキーの値をテストする条件の作成」を参照してください。

このポリシーは、リクエストで指定されたすべてのタグキーが承認されたタグキーであることを保証します。条件の ForAnyValue 修飾子によって、指定したキーの少なくとも 1 つがリクエストに存在することが保証されます。

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
            "arn:aws:iam::111122223333:role/JohnDoe"
         ]
       },
 "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"ForAnyValue:StringEquals": {"s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
    }
  ]
}
```

ユーザーにオブジェクトタグの追加を許可する場合は特定のタグキーと値が必要

次のポリシーの例では、s3:PutObjectTagging アクションを実行するアクセス許可をユーザーに付与します (ユーザーが既存のオブジェクトにタグを追加することができます)。この条件により、値が *X* に設定された特定のタグキー (*Project* など) をユーザーが含めることが求められます。

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
       "arn:aws:iam::111122223333:user/JohnDoe"
         ]
       },
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
```

```
      ],
      "Condition": {"StringEquals": {"s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

特定のオブジェクトタグキーと値を持つオブジェクトのみを追加することをユーザーに許可する

次のポリシーの例では、s3:PutObject アクションを実行する権限をユーザーに付与して、ユーザーがバケットにオブジェクトを追加できるようにします。ただし、Condition ステートメントは、アップロードされたオブジェクトで使用できるタグキーと値を制限します。この例では、ユーザーがバケットに追加できるのは、値が *Finance* に設定された特定のタグキー (*Department*) を持つオブジェクトだけです。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Principal":{
            "AWS":[
                 "arn:aws:iam::111122223333:user/JohnDoe"
         ]
        },
        "Effect": "Allow",
        "Action": [
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        ],
        "Condition": {
            "StringEquals": {
                "s3:RequestObjectTag/Department": "Finance"
            }
        }
    }]
}
```

グローバル条件キーによるオブジェクトアクセスの管理

グローバル条件キーは、aws プレフィックスが付いた条件キーです。AWS のサービス のサービスは、グローバル条件キーをサポートするか、サービスプレフィックスを含むサービス固有のキーを提供できます。JSON ポリシーの Condition 要素を使用して、リクエストのキーを、ポリシーで指定したキー値と比較できます。

Amazon S3 サーバーアクセスログ配信のみに制限する

次の例のバケットポリシーでは、aws:SourceArn グローバル条件キーを使用して、サービス間リクエストを行っているリソースの Amazon リソースネーム (ARN) を、ポリシーで指定した ARN と比較します。aws:SourceArn 条件キーを使用して、サービス間のトランザクション中に Amazon S3 サービスが混乱した代理として使用されるのを防ぐことができます。Amazon S3 バケットにオブジェクトを追加できるのは Amazon S3 サービスのみです。

この例のバケットポリシーは、ロギングサービスプリンシパル (s3:PutObject) にのみ logging.s3.amazonaws.com 許可を付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutObjectS3ServerAccessLogsPolicy",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET-logs/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111111111111"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::EXAMPLE-SOURCE-BUCKET"
                }
            }
        },
        {
            "Sid": "RestrictToS3ServerAccessLogs",
            "Effect": "Deny",
            "Principal": "*",
```

```
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET-logs/*",
            "Condition": {
                "ForAllValues:StringNotEquals": {
                    "aws:PrincipalServiceNamesList": "logging.s3.amazonaws.com"
                }
            }
        }
    ]
}
```

自分の組織にのみアクセスを許可

リソースにアクセスするすべての IAM プリンシパルが組織内の AWS アカウント (AWS Organizations管理アカウントを含む) からのアクセスのみに制限する場合は、aws:PrincipalOrgID グローバル条件キーを使用できます。

このタイプのアクセスを許可または制限するには、aws:PrincipalOrgID 条件を定義し、バケットポリシーの組織 ID に値を設定します。組織 ID は、バケットへのアクセスを制御するために使用されます。aws:PrincipalOrgID 条件を使用すると、バケットポリシーのアクセス許可は、組織に追加されるすべての新しいアカウントにも適用されます。

以下は、組織内の特定の IAM プリンシパルにバケットへの直接アクセス許可を付与できるリソースベースのバケットポリシーの例です。バケットポリシーに aws:PrincipalOrgID グローバル条件キーを追加すると、リソースにアクセスするにはプリンシパルアカウントが組織内に存在する必要があります。アクセスを許可するときに誤って間違ったアカウントを指定した場合でも、aws:PrincipalOrgID グローバル条件キーは追加の保護手段として機能します。このグローバルキーをポリシーで使用すると、指定した組織以外のすべてのプリンシパルが、Amazon S3 バケットにアクセスできないようにします。リソースへのアクセス権を取得できるのは、リストにある組織のアカウントのプリンシパルだけです。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowGetObject",
        "Principal": {
            "AWS": "*"
        },
        "Effect": "Allow",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
```

```
        "Condition": {
            "StringEquals": {
                "aws:PrincipalOrgID": ["o-aa111bb222"]
            }
        }
    }]
}
```

特定の IP アドレスに基づくアクセス管理

特定の IP アドレスへのアクセスの制限

以下の例では、リクエストが指定した IP アドレス範囲からのものでない限り、指定したバケット内のオブジェクトに対してユーザーが Amazon S3 のオペレーションを実行できないようにします。

> **Note**
>
> 特定の IP アドレスへのアクセスを制限する場合は、S3 バケットにアクセスできる VPC エンドポイント、VPC ソース IP アドレス、または外部 IP アドレスも必ず指定してください。そうしないと、適切な権限がまだ設定されていない限り、すべてのユーザーがバケット内のオブジェクトに対して S3 オペレーションを実行することをポリシーで拒否すると、バケットにアクセスできなくなる可能性があります。

このポリシーの `Condition` ステートメントは、許可されたインターネットプロトコルバージョン 4 (IPv4) の IP アドレスの範囲として、*`192.0.2.0/24`* を識別します。

`Condition` ブロックでは、`NotIpAddress` 条件と `aws:SourceIp` 条件キー (AWS 全体をターゲットとする条件キー) を使用します。`aws:SourceIp` 条件キーは、パブリック IP アドレス範囲にのみ使用できます。これらの条件キーの詳細については、Amazon S3 のポリシー条件キー を参照してください。`aws:SourceIp` IPv4 値は標準の CIDR 表記を使用します。詳細については、IAM ユーザーガイドの「IAM JSON ポリシー要素のリファレンス」を参照してください。

> **Warning**
>
> このポリシーを使用する前に、この例の *`192.0.2.0/24`* IP アドレス範囲をユースケースに適した値に置き換えてください。置き換えないと、バケットにアクセスできなくなります。

```
{
```

```
    "Version": "2012-10-17",
    "Id": "S3PolicyId1",
    "Statement": [
        {
            "Sid": "IPAllow",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ],
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "192.0.2.0/24"
                }
            }
        }
    ]
}
```

IPv4 アドレスと IPv6 アドレスの許可

IPv6 アドレスの使用を開始する場合は、既存の IPv4 アドレス範囲に IPv6 アドレス範囲を追加して組織のすべてのポリシーを更新することをお勧めします。こうすることで、IPv6 への移行後もポリシーが引き続き機能するようになります。

以下のバケットポリシーの例は、組織の有効な IP アドレスすべてを含めるために、IPv4 アドレス範囲と IPv6 アドレス範囲を混在させる方法を示しています。このポリシーの例では、サンプル IP アドレス (*192.0.2.1* および *2001:DB8:1234:5678::1*) へのアクセスを許可したり、アドレス *203.0.113.1* および *2001:DB8:1234:5678:ABCD::1* へのアクセスを拒否したりできます。

`aws:SourceIp` 条件キーは、パブリック IP アドレス範囲にのみ使用できます。`aws:SourceIp` の IPv6 の値は、標準の CIDR 形式で指定する必要があります。IPv6 では、0 の範囲を表すために `::` の使用がサポートされています (例: `2001:DB8:1234:5678::/64`)。詳細については、IAM ユーザーガイドの IP アドレス条件演算子 を参照してください。

> **Warning**
>
> このポリシーを使用する前に、この例の IP アドレス範囲をユースケースに適した値に置き換えます。置き換えないと、バケットにアクセスできなくなる可能性があります。

```
{
    "Id": "PolicyId2",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIPmix",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "2001:DB8:1234:5678::/64"
                    ]
                },
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "203.0.113.0/24",
                        "2001:DB8:1234:5678:ABCD::/80"
                    ]
                }
            }
        }
    ]
}
```

HTTP または HTTPS リクエストに基づくアクセス管理

HTTPS リクエストのみにアクセスを制限

潜在的な攻撃者がネットワークトラフィックを操作するのを防ぎたい場合は、HTTPS (TLS) を使用して暗号化された接続のみを許可し、HTTP リクエストによるバケットへのアクセスを制限できます。リクエストが HTTP か HTTPS かを判断するには、S3 バケットポリシーの aws:SecureTransport グローバル条件キーを使用します。aws:SecureTransport 条件キーは、リクエストが HTTP を使用して送信されたかどうかをチェックします。

リクエストが true を返した場合、リクエストは HTTPS 経由の送信です。リクエストが false を返した場合、リクエストは HTTP 経由の送信です。その後、目的のリクエストスキームに基づいてバケットへのアクセスを許可または拒否することができます。

次の例で、バケットポリシーは HTTP リクエストを明示的に拒否しています。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "RestrictToTLSRequestsOnly",
        "Action": "s3:*",
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        ],
        "Condition": {
            "Bool": {
                "aws:SecureTransport": "false"
            }
        },
        "Principal": "*"
    }]
}
```

特定の HTTP Referer へのアクセスの制限

example-s3-bucket というバケットに格納されている写真や動画へのリンクがある *www.example.com* または *example.com* というドメイン名のウェブサイトがあるとします。デフォルトでは、Amazon S3 のすべてのリソースはプライベートであるため、リソースを作成した AWS アカウント のみがアクセスできます。

これらのオブジェクトへのウェブサイトの読み取りアクセスを許可するには、s3:GetObject アクセス許可を条件付きで付与するバケットポリシーを追加する方法があります。条件としては、GET リクエストが特定のウェブページから発生する必要があることを指定します。次のポリシーでは、aws:Referer 条件キー付きの StringLike 条件を使用してリクエストを制限します。

```
{
  "Version":"2012-10-17",
  "Id":"HTTP referer policy example",
  "Statement":[
    {
```

```
      "Sid":"Allow only GET requests originating from www.example.com and
 example.com.",
      "Effect":"Allow",
      "Principal":"*",
      "Action":["s3:GetObject","s3:GetObjectVersion"],
      "Resource":"arn:aws:s3:::example-s3-bucket/*",
      "Condition":{
        "StringLike":{"aws:Referer":["http://www.example.com/*","http://example.com/
*"]}
      }
    }
  ]
}
```

使用するブラウザのリクエストに HTTP `referer` ヘッダーが含まれていることを確認します。

> **Warning**
>
> `aws:Referer` 条件キーを使用するときには、十分な注意が必要です。一般に知られている HTTP 参照子のヘッダー値を含めるのは危険です。不正な当事者は、変更されたブラウザまたはカスタムブラウザを使用して任意の `aws:Referer` 値を提供することができます。したがって、無許可の当事者が AWS リクエストを直接作成できないよう、`aws:Referer` を使用しないでください。
>
> この `aws:Referer` 条件キーは、Amazon S3 に保存されているコンテンツなどのデジタルコンテンツが、無許可のサードパーティーサイトで参照されないよう保護する目的でのみ、お客様に提供されています。詳細については、IAM ユーザーガイドaws:Refererの を参照してください。

特定のフォルダへのユーザーアクセスの管理

特定のフォルダへのアクセス許可をユーザーに付与

特定のフォルダへのアクセスをユーザーに許可しようとしているとします。IAM ユーザーと S3 バケットが同じ AWS アカウント に属している場合は、IAM ポリシーを使用してユーザーに特定のバケットフォルダへのアクセス許可を付与できます。このアプローチを使用すると、バケットポリシーを更新してアクセスを付与する必要はありません。複数のユーザーが切り替えることができる IAM ロールに IAM ポリシーを追加できます。

IAM ID と S3 バケットの AWS アカウント が異なる場合は、IAM ポリシーとバケットポリシーの両方でクロスアカウントアクセスを許可する必要があります。クロスアカウントアクセスを付与する方

法の詳細については、「バケット所有者がクロスアカウントのバケットのアクセス許可を付与する」を参照してください。

次のバケットポリシーの例では、自分のフォルダ (home/*JohnDoe*/) のみに*JohnDoe* のフルコンソールアクセスを許可しています。home フォルダを作成し、ユーザーに適切なアクセス許可を付与することで、複数のユーザーが 1 つのバケットを共有できます。このポリシーは、次の 3 つの Allow ステートメントで構成されています。

- *AllowRootAndHomeListingOfCompanyBucket*: ユーザー (*JohnDoe*) が *DOC-EXAMPLE-BUCKET* バケットのルートレベルと home フォルダ内のオブジェクトを一覧表示できるようにします。このステートメントにより、ユーザーはコンソールを使用してプレフィックス home/ を検索することもできます。
- *AllowListingOfUserFolder*: ユーザー (*JohnDoe*) が home/*JohnDoe*/ フォルダとサブフォルダ内のすべてのオブジェクトを一覧表示できるようにします。
- *AllowAllS3ActionsInUserFolder*: Read、Write、Delete 権限を付与することで、ユーザーが Amazon S3 のすべてのアクションを実行できるようにします。権限はバケット所有者のホームフォルダに限定されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRootAndHomeListingOfCompanyBucket",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Effect": "Allow",
            "Action": ["s3:ListBucket"],
            "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
            "Condition": {
                "StringEquals": {
                    "s3:prefix": ["", "home/", "home/JohnDoe"],
                    "s3:delimiter": ["/"]
                }
            }
        },
        {
            "Sid": "AllowListingOfUserFolder",
```

```
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Action": ["s3:ListBucket"],
            "Effect": "Allow",
            "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
            "Condition": {
                "StringLike": {
                    "s3:prefix": ["home/JohnDoe/*"]
                }
            }
        },
        {
            "Sid": "AllowAllS3ActionsInUserFolder",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Action": ["s3:*"],
            "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/home/JohnDoe/*"]
        }
    ]
}
```

アクセスログへのアクセス管理

アクセスログを有効にするためのアクセス権限を Application Load Balancer に付与

Application Load Balancer のアクセスログを有効にする場合は、ロードバランサーがログを保存する S3 バケットの名前を指定する必要があります。このバケットは、バケットにアクセスログを書き込む許可を Elastic Load Balancing に付与するアタッチされたポリシーが必要です。

次の例では、バケットポリシーにより、バケットにアクセスログを書き込む許可を Elastic Load Balancing (ELB) に付与しています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Principal": {
        "AWS": "arn:aws:iam::elb-account-id:root"
      },
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::example-s3-bucket/prefix/AWSLogs/111122223333/*"
    }
  ]
}
```

> **Note**
>
> 必ず、*elb-account-id* を、ご利用の AWS リージョン における Elastic Load Balancing の AWS アカウント ID に置き換えてください。Elastic Load Balancing リージョンのリストについては、「Elastic Load Balancing ユーザーガイド」の「Amazon S3 バケットにポリシーをアタッチする」を参照してください。

ご利用の AWS リージョン がサポートされている Elastic Load Balancing リージョンのリストに表示されない場合は、次のポリシーを使用して、指定されたログ配信サービスにアクセス許可を付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
       "Principal": {
         "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
          },
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::example-s3-bucket/prefix/AWSLogs/111122223333/*"
    }
  ]
}
```

次に、必ず Elastic Load Balancing のアクセスログを有効にして設定してください。テストファイルを作成することで、バケットのアクセス許可を確認できます。

Amazon CloudFront の OAI へのアクセス管理

Amazon CloudFront の OAI へのアクセス許可の付与

次のバケットポリシーの例は、CloudFront のオリジンアクセスアイデンティティ (OAI) に S3 のバケット内のすべてのオブジェクトを取得 (読み取り) するアクセス許可を付与します。CloudFront の OAI を使用すると、バケット内のオブジェクトへの CloudFront からのアクセスは許可して、Amazon S3 からの直接アクセスは許可しないようにすることができます。詳細については、「Amazon CloudFront デベロッパーガイド」の「オリジンアクセス ID を使用して Amazon S3 コンテンツへのアクセスを制限する」を参照してください。

次のポリシーは、OAI の ID をポリシーの `Principal` として使用します。S3 のバケットポリシーを使用して CloudFront の OAI にアクセス許可を付与する方法の詳細については、「Amazon CloudFront デベロッパーガイド」の「オリジンアクセスアイデンティティ (OAI) からオリジンアクセスコントロール (OAC) への移行」を参照してください。

この例を使用するには:

- *EH1HDMB1FH2TC* を OAI の ID に置き換えます。OAI の ID を確認するには、CloudFront コンソールのオリジンアクセスアイデンティティページを参照するか、CloudFront API の ListCloudFrontOriginAccessIdentities を使用します。
- *example-s3-bucket* をバケットの名前に置き換えます。

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity EH1HDMB1FH2TC"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::example-s3-bucket/*"
        }
    ]
}
```

Amazon S3 ストレージレンズへのアクセス管理

Amazon S3 ストレージレンズへのアクセス許可の付与

S3 ストレージレンズはメトリクスを集約し、Amazon S3 コンソールの [Buckets] (バケット) ページの [Account snapshot] (アカウントスナップショット) セクションにこの情報を表示します。S3 Storage Lens は、インサイトと傾向を可視化したり、外れ値にフラグ付けしたり、ストレージコストの最適化やデータ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取ったりするために使用できるインタラクティブダッシュボードも提供します。ダッシュボードには、組織、アカウント、AWS リージョン、バケット、オブジェクト、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成して可視化できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。

S3 ストレージレンズは、収集したストレージ使用量のメトリクスをさらなる分析のため Amazon S3 バケット内にエクスポートできます。S3 Storage Lens がメトリクスのエクスポートを配置するバケットは、送信先バケットとして知られています。S3 Storage Lens のメトリクスエクスポートを設定するとき、ターゲットバケットにバケットポリシーを作成する必要があります。詳細については、「Amazon S3 ストレージレンズを使用してストレージのアクティビティと使用状況を評価する」を参照してください。

次のバケットポリシーの例では、Amazon S3 が送信先バケットにオブジェクトを書き込む (PUT リクエスト) ためのアクセス許可が付与されます。S3 Storage Lens メトリクスのエクスポートを設定するときには、このようなバケットポリシーを送信先バケットに使用します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3StorageLensExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "storage-lens.s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::destination-bucket/destination-prefix/
StorageLens/111122223333/*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn": "arn:aws:s3:region-code:111122223333:storage-
lens/storage-lens-dashboard-configuration-id"
                }
            }
        }
    ]
}
```

S3 ストレージレンズの組織レベルのメトリクスエクスポートを設定するときには、前述のバケットポリシーの Resource ステートメントに次の変更を加えます。

```
"Resource": "arn:aws:s3:::destination-bucket/destination-prefix/StorageLens/your-
organization-id/*",
```

S3 インベントリ、S3 分析、および S3 インベントリレポートの権限管理

S3 インベントリおよび S3 分析に対するアクセス許可の付与

S3 インベントリでは、バケット内のオブジェクトのリストが作成され、S3 分析のエクスポートでは、分析に使用されるデータの出力ファイルが作成されます。インベントリによってオブジェクトがリストされるバケットは、ソースバケットと呼ばれます。インベントリファイルと、分析エクスポートファイルが書き込まれるバケットは、ターゲットバケットと呼ばれます。インベントリまたは分析のエクスポートを設定する場合、ターゲットバケットにバケットポリシーを作成する必要があります。詳細については、Amazon S3 インベントリおよびAmazon S3 分析 – ストレージクラス分析を参照してください。

次のバケットポリシーの例では、ソースバケットのアカウントからターゲットバケットにオブジェクトを書き込む (PUT リクエスト) ための Amazon S3 のアクセス許可が付与されます。このようなバケットポリシーは、S3 インベントリと S3 分析エクスポートをセットアップするときに、宛先バケットで使用します。

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
            "Sid": "InventoryAndAnalyticsExamplePolicy",
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-DESTINATION-BUCKET/*"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET"
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

S3 インベントリレポート設定の作成を制御する

Amazon S3 インベントリ は、S3 バケット内のオブジェクトのリストと、各オブジェクトのメタデータを作成します。s3:PutInventoryConfiguration アクセス許可により、ユーザーはデフォルトで使用可能なすべてのオブジェクトメタデータフィールドを含むインベント設定を作成し、インベントリを保存する宛先バケットを指定できます。宛先バケット内のオブジェクトへの読み取りアクセスを持つユーザーは、インベントリレポートで使用可能なすべてのオブジェクトメタデータフィールドにアクセスできます。S3 Inventory で使用できるメタデータフィールドの詳細については、「Amazon S3 インベントリのリスト」を参照してください。

S3 インベントリレポートをユーザーが設定できないようにするには、ユーザーから s3:PutInventoryConfiguration アクセス許可を削除します。

S3 インベントリレポート設定の一部のオブジェクトメタデータフィールドはオプションです。つまり、デフォルトで使用できますが、ユーザーにs3:PutInventoryConfiguration アクセス許可を付与すると制限できます。s3:InventoryAccessibleOptionalFields 条件キーを使用して、ユーザーがこれらのオプションのメタデータフィールドをレポートに含めることができるかどうかを制御できます。S3 インベントリで使用できるオプションのメタデータフィールドのリストについては、「Amazon Simple Storage Service API リファレンス」の「OptionalFields」を参照してください。

特定のオプションのメタデータフィールドを使用してインベントリ設定を作成するアクセス許可をユーザーに付与するには、s3:InventoryAccessibleOptionalFields 条件キーを使用してバケットポリシーの条件を絞り込みます。

次のポリシー例では、インベントリ設定を条件付きで作成するアクセス許可をユーザー (*Ana*) に付与します。ポリシーの ForAllValues:StringEquals条件は、s3:InventoryAccessibleOptionalFields条件キーを使用して、許可される 2 つのオプションのメタデータフィールド、つまり Size と StorageClass を指定します。したがって、*Ana* がインベントリ設定を作成するとき、含めることができるオプションのメタデータフィールドは Size と StorageClass のみです。

```
{
 "Id": "InventoryConfigPolicy",
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "AllowInventoryCreationConditionally",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action":
    "s3:PutInventoryConfiguration",
   "Resource":
    "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
   "Condition": {
    "ForAllValues:StringEquals": {
     "s3:InventoryAccessibleOptionalFields": [
        "Size",
        "StorageClass"
        ]
      }
    }
   }
 ]
}
```

特定のオプションのメタデータフィールドを含む S3 インベントリレポートをユーザーが設定できないようにするには、レプリケート元バケットのバケットポリシーに明示的な Deny ステートメントを追加します。次のバケットポリシーの例では、オプションの ObjectAccessControlList または ObjectOwner メタデータフィールドを含むインベントリ設定をソースバケット *DOC-EXAMPLE-*

SOURCE-BUCKET に作成することをユーザー *Ana* に拒否します。ユーザー*Ana* は、他のオプションのメタデータフィールドを使用してインベントリ設定を作成できます。

```
{
 "Id": "InventoryConfigSomeFields",
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "AllowInventoryCreation",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action": "s3:PutInventoryConfiguration",
   "Resource":
    "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",

  },
  {
   "Sid": "DenyCertainInventoryFieldCreation",
   "Effect": "Deny",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action": "s3:PutInventoryConfiguration",
   "Resource":
     "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
   "Condition": {
    "ForAnyValue:StringEquals": {
     "s3:InventoryAccessibleOptionalFields": [
        "ObjectOwner",
        "ObjectAccessControlList"
        ]
      }
    }
   }
 ]
}
```

> **Note**
>
> バケットポリシーで `s3:InventoryAccessibleOptionalFields` 条件キーを使用しても、既存のインベントリ設定に基づくインベントリレポートのデリバリーには影響しません。

> **Important**
>
> 前の例に示すように、`Allow` 効果で `ForAllValues` を、または `Deny` 効果で `ForAnyValue` を使用することをお勧めします。
> これらの組み合わせは過度に制限され、インベントリ設定の削除がブロックされる可能性があるため、`Deny` 効果で `ForAllValues` または `Allow` 効果で `ForAnyValue` を使用しないでください。
> `ForAllValues` および `ForAnyValue` 条件セット演算子の詳細については、「IAM ユーザーガイド」「複数値のコンテキストキー」を参照してください。

MFA が必要

Amazon S3 は、多工レメント認証 (MFA) で保護された API へのアクセスをサポートしています。この機能により、Amazon S3 のリソースへのアクセスに MFA を強制的に適用することができます。多工レメント認証により、AWS 環境に適用できるセキュリティのレベルが高まります。MFA は、有効な MFA コードを入力して MFA デバイスを物理的に所有していることを証明することがユーザーに要求されるセキュリティ機能です。詳細については、AWS 多工レメント認証 を参照してください。Amazon S3 のリソースにアクセスするすべてのリクエストに対して MFA を要求することができます。

MFA の要件を適用するには、バケットポリシーで `aws:MultiFactorAuthAge` 条件キーを使用します。IAM ユーザーは、AWS Security Token Service (AWS STS) により発行される一時的な認証情報を使用して、Amazon S3 のリソースにアクセスできます。AWS STS リクエスト時に、MFA コードを指定します。

Amazon S3 が多要素認証のリクエストを受け取ると、`aws:MultiFactorAuthAge` 条件キーに一時的な認証情報が作成されてからの時間の数値 (秒) が示されます。リクエストで提供された一時的な認証情報が MFA デバイスを使用して作成されていない場合、このキー値は null (不在) になります。次の例に示すように、バケットポリシーに、この値を確認する条件を追加できます。

このポリシー例は、リクエストが MFA を使用して認証されていない場合、*example-s3-bucket* バケットの */taxdocuments* フォルダに対するすべての Amazon S3 オペレーションを拒否します。MFA の詳細については、IAM ユーザーガイドAWSの での多工レメント認証 (MFA) の使用 を参照してください。

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::example-s3-bucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true }}
      }
    ]
 }
```

aws:MultiFactorAuthAge 条件キー値が null で、リクエスト内の一時的なセキュリティ認証情報が MFA デバイスを使用せずに作成されたことを示している場合、Condition ブロック内の Null 条件の評価は true になります。

次のバケットポリシーは、前述のバケットポリシーの拡張です。次のバケットポリシーには、2 つのポリシーステートメントが含まれています。1 つのステートメントは、バケット (*example-s3-bucket*) の s3:GetObject アクセス許可を全員に付与します。もう 1 つのステートメントは、MFA を要求することにより、バケットの *example-s3-bucket/taxdocuments* フォルダへのアクセスを制限します。

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::example-s3-bucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true } }
```

```
      },
      {
        "Sid": "",
        "Effect": "Allow",
        "Principal": "*",
        "Action": ["s3:GetObject"],
        "Resource": "arn:aws:s3:::example-s3-bucket/*"
      }
    ]
 }
```

オプションで、aws:MultiFactorAuthAge キーの有効期間を制限する数値条件を使用することができます。aws:MultiFactorAuthAge キーで指定する期間は、リクエストの認証に使われる一時的セキュリティ認証情報の寿命とは無関係です。

例えば、次のバケットポリシーでは、MFA 認証を要求するほかに、一時セッションが作成されてからの時間もチェックします。このポリシーは、aws:MultiFactorAuthAge キーの値が、一時セッションが 1 時間 (3,600 秒) 以上前に作成されたことを示す場合に、すべてのオペレーションを拒否します。

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::example-s3-bucket/taxdocuments/*",
        "Condition": {"Null": {"aws:MultiFactorAuthAge": true }}
      },
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::example-s3-bucket/taxdocuments/*",
        "Condition": {"NumericGreaterThan": {"aws:MultiFactorAuthAge": 3600 }}
       },
       {
         "Sid": "",
         "Effect": "Allow",
```

```
        "Principal": "*",
        "Action": ["s3:GetObject"],
        "Resource": "arn:aws:s3:::example-s3-bucket/*"
      }
    ]
}
```

ユーザーによるオブジェクトの削除を禁止する

デフォルトでは、ユーザーには一切のアクセス許可がありません。ただし、ポリシーを作成する際に、意図しないアクセス許可をユーザーに付与する可能性があります。このようなアクセス許可の抜け穴を防ぐには、明示的な拒否を追加して、より厳格なアクセスポリシーを記述する必要があります。

ユーザーまたはアカウントによるオブジェクトの削除を明示的に防ぐには、バケットポリシーに `s3:DeleteObject`、`s3:DeleteObjectVersion`、および `s3:PutLifecycleConfiguration` のアクションのアクセス許可を追加する必要があります。明示的に DELETE Object API を呼び出すか、有効期限が過ぎたオブジェクトを Amazon S3 が削除できるようにオブジェクトのライフサイクルを設定 (「ストレージのライフサイクルの管理」を参照) してオブジェクトを削除するため、これらすべてのアクションが必要です。

次の例は、ユーザー Dave の DELETE Object のアクセス許可を明示的に拒否します。明示的な拒否は、付与されている他のアクセス許可に常に優先されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/Dave"
      },
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetBucketAcl"
      ],
      "Resource": [
        "arn:aws:s3:::example-s3-bucket1",
   "arn:aws:s3:::example-s3-bucket1/*"
      ]
    },
```

```
    {
      "Sid": "statement2",
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/Dave"
      },
      "Action": [
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutLifecycleConfiguration"
      ],
      "Resource": [
        "arn:aws:s3:::example-s3-bucket1",
     "arn:aws:s3:::example-s3-bucket1/*"
      ]
    }
  ]
}
```

条件キーを使用したバケットポリシーの例

アクセスポリシー言語を使用して、アクセス許可を付与するときに条件を指定できます。オプションの Condition エレメント、または Condition ブロックを使用して、ポリシーが有効になるときの条件を指定できます。

オブジェクトおよびバケットオペレーションに Amazon S3 条件キーを使用するポリシーについては、次の例を参照してください。条件キーの詳細については、Amazon S3 のポリシー条件キー を参照してください。Amazon S3 アクションの完全なリストとポリシーで指定できるリソースについては、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

例 — オブジェクトオペレーションで使用できる Amazon S3 の条件キー

このセクションでは、オブジェクトオペレーションに Amazon S3 固有の条件キーを使用する方法の例を示します。Amazon S3 アクションの完全なリストとポリシーで指定できるリソースについては、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

ここでは、PutObject オペレーションで条件キーを使用するポリシーの例をいくつか示します。PutObject オペレーションでは、アクセスコントロールリスト (ACL) に固有のヘッダーを使用して、ACL に基づいてアクセス許可を付与することができます。バケット所有者は、これらのキーを

使用して条件を設定し、ユーザーがオブジェクトをアップロードする場合に特定のアクセス許可を要求することができます。また、PutObjectAcl オペレーションを使用して ACL に基づいてアクセス許可を付与することもできます。詳細については、Amazon S3 Amazon Simple Storage Service API リファレンスの PutObjectAcl を参照してください。ACL の詳細については、アクセスコントロールリスト (ACL) の概要 を参照してください。

トピック

例 1: サーバー側の暗号化を使用してオブジェクトを保存するよう要求する s3:PutObject のアクセス許可を付与する

アカウント A がバケット所有者であるとします。アカウント管理者がアカウント A のユーザーの Jane にオブジェクトをアップロードするアクセス許可を付与するとします。ただし、必ずサーバー側の暗号化をリクエストすることを条件とします。これにより、Amazon S3 でオブジェクトが暗号化されて保存されます。この場合、アカウント A の管理者は、以下のように `s3:x-amz-server-side-encryption` 条件キーを使用することができます。`Condition` ブロックのキーと値のペアは、`s3:x-amz-server-side-encryption` キーを指定します。

```
"Condition": {
     "StringNotEquals": {
         "s3:x-amz-server-side-encryption": "AES256"
     }}
```

AWS CLI を使用してアクセス許可をテストする場合は、`--server-side-encryption` パラメータを使用して、必須パラメータを追加する必要があります。

```
aws s3api put-object --bucket example1bucket --key HappyFace.jpg --body c:
\HappyFace.jpg --server-side-encryption "AES256" --profile AccountBadmin
```

例 2: 制限されているコピー元からオブジェクトをコピーする s3:PutObject のアクセス許可を付与する

PUT Object のリクエストでは、ソースのオブジェクトを指定すると、コピーオペレーションを実行できます (「PUT Object - Copy」を参照)。バケット所有者は、これに対応して、ソースに制限が付いているオブジェクトをコピーするためのアクセス許可をユーザーに付与できます。以下に例を示します。

- sourcebucket バケットからのみオブジェクトをコピーすることを許可します。
- ソースバケットのオブジェクトのコピーを許可し、キー名のプレフィックスが public/ で始まるオブジェクト (例: sourcebucket/public/* など) のみを許可します。
- ソースバケットの特定のオブジェクト (例: sourcebucket/example.jpg など) のコピーのみを許可します。

次のバケットポリシーでは、ユーザー (Dave) に s3:PutObject のアクセス許可が付与されます。Dave は、リクエストに s3:x-amz-copy-source ヘッダーを含め、ヘッダーの値でキー名のプレフィックスに /awsexamplebucket1/public/* を指定している場合にのみ、オブジェクトをコピーすることができます。

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Sid": "cross-account permission to user in your own account",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*"
        },
        {
            "Sid": "Deny your user permission to upload object if copy source is not /
bucket/folder",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
```

```
            "Condition": {
                "StringNotLike": {
                    "s3:x-amz-copy-source": "awsexamplebucket1/public/*"
                }
            }
        }
    ]
}
```

AWS CLI でポリシーをテストする

AWS CLI の copy-object コマンドを使用して、アクセス許可をテストできます。--copy-source パラメータを追加してソースを指定します。キー名のプレフィックスは、ポリシーで許可されているプレフィックスと一致する必要があります。--profile パラメータを使用して、ユーザー Dave の認証情報を指定します。AWS CLI のセットアップの詳細については、AWS CLI を使用した Amazon S3 での開発 を参照してください。

```
aws s3api copy-object --bucket awsexamplebucket1 --key HappyFace.jpg
--copy-source examplebucket/public/PublicHappyFace1.jpg --profile AccountADave
```

特定のオブジェクトのみをコピーするアクセス許可を付与する

前述のポリシーでは、StringNotLike 条件を使用しています。特定のオブジェクトのみをコピーするアクセス許可を付与するには、条件を StringNotLike から StringNotEquals に変更し、次のようにオブジェクトキーを正確に指定する必要があります。

```
"Condition": {
       "StringNotEquals": {
           "s3:x-amz-copy-source": "awsexamplebucket1/public/PublicHappyFace1.jpg"
       }
}
```

例 3: オブジェクトの特定のバージョンに対するアクセス許可を付与する

アカウント A が、バージョンを使用可能なバケットを所有しているとします。このバケットには、HappyFace.jpg オブジェクトのバージョンが複数含まれています。アカウント管理者は、ユーザー Dave にオブジェクトの特定のバージョンのみを取得できるアクセス許可を付与したいと考えています。この場合、アカウント管理者は、次に示すように、条件を付けて Dave に s3:GetObjectVersion のアクセス許可を付与できます。Condition ブロックのキーと値のペアは、s3:VersionId 条件キーを指定します。この場合、Dave がオブジェクトを取得するには、オブジェクトのバージョン ID を正確に知っている必要があります。

詳細については、Amazon Simple Storage Service API リファレンスの GetObject を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::examplebucketversionenabled/HappyFace.jpg"
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::examplebucketversionenabled/HappyFace.jpg",
            "Condition": {
                "StringNotEquals": {
                    "s3:VersionId": "AaaHbAQitwiL_h47_44lRO2DDfLlBO5e"
                }
            }
        }
    ]
}
```

AWS CLI でポリシーをテストする

このアクセス許可をテストするには、AWS CLI の `get-object` コマンドを実行するときに、特定のオブジェクトバージョンを指定する `--version-id` パラメータを使用します。コマンドは、オブジェクトを取得し、`OutputFile.jpg` ファイルに保存します。

```
aws s3api get-object --bucket examplebucketversionenabled --key HappyFace.jpg
 OutputFile.jpg --version-id AaaHbAQitwiL_h47_44lRO2DDfLlBO5e --profile AccountADave
```

例 4: オブジェクトタグに基づくアクセス許可の付与

Amazon S3 のオペレーションでオブジェクトのタグの条件キーを使用する方法の例については、タグ付けとアクセスコントロールポリシー を参照してください。

例 5: バケット所有者の AWS アカウント ID によるアクセスの制限

`aws:ResourceAccount` キーと `s3:ResourceAccount` キーのいずれかを使用して、特定の AWS アカウント ID が所有する Amazon S3 バケットへのユーザーまたはアプリケーションのアクセスを制限する IAM または仮想プライベートクラウド (VPC) エンドポイントポリシーを記述できます。この条件キーを使用して、VPC 内のクライアントが、ユーザーが所有していないバケットにアクセスすることを制限できます。

ただし、一部の AWS サービスは AWS マネージドバケットへのアクセスに依存していることに注意してください。したがって、IAM ポリシーで `aws:ResourceAccount` キーまたは `s3:ResourceAccount` キーを使用することは、これらのリソースへのアクセスにも影響する可能性があります。

詳細および例については、次のリソース を参照してください。

- AWS アカウント ガイドの指定された AWS PrivateLink のバケットへのアクセスの制限
- Amazon ECR ガイドの Amazon ECR が使用するバケットへのアクセスを制限する
- AWS Systems Manager ガイドの AWS が管理する Amazon S3 バケットに必要なアクセスを System Manager に提供する
- AWS ストレージブログの Limit access to owned by specific AWS アカウント

例 6: 最小の TLS バージョンの要求

s3:TlsVersion 条件キーを使用して、単純な IAM、Virtual Private Cloud Endpoint (VPCE)、またはバケットポリシーを記述し、クライアントが使用する TLS バージョンに基づいて Amazon S3 バケットへのユーザーまたはアプリケーションのアクセスを制限できます。この条件キーを使用して、最小の TLS バージョンを必要とするポリシーを記述できます。

Example

このバケットポリシーの例では、1.2 よりも低い TLS バージョン (1.1 や 1.0 など) のクライアントによる PutObject リクエストを拒否します。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::example-s3-bucket1",
                "arn:aws:s3:::example-s3-bucket1/*"
            ],
            "Condition": {
                "NumericLessThan": {
                    "s3:TlsVersion": 1.2
                }
            }
        }
    ]
}
```

Example

このバケットポリシー例では、1.1 よりも高い TLS バージョン (1.2、1.3 以上など) のクライアントによる PutObject リクエストを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::example-s3-bucket1",
                "arn:aws:s3:::example-s3-bucket1/*"
            ],
            "Condition": {
                "NumericGreaterThan": {
                    "s3:TlsVersion": 1.1
                }
            }
        }
```

```
    ]
}
```

例 — バケットオペレーションで使用できる Amazon S3 の条件キー

このセクションでは、バケットオペレーションに Amazon S3 の固有の条件キーを使用するポリシーの例を示します。

トピック

- 例 1: IP アドレスに対する条件付きの s3:GetObject アクセス許可の付与
- 例 2: 特定のプレフィックスを持つバケット内のオブジェクト一覧の取得
- 例 3: キーの最大数の設定

例 1: IP アドレスに対する条件付きの s3:GetObject アクセス許可の付与

リクエストが特定の範囲の IP アドレス (192.0.2.0.*、ただし 192.0.2.188 を除く) から発信された場合、認証されたユーザーに `s3:GetObject` アクションを使用するアクセス許可を付与することができます。条件ブロックの `IpAddress` と `NotIpAddress` は条件であり、それぞれの条件では評価されるキーと値のペアが指定されています。この例の両方のキーと値のペアでは、`aws:SourceIp` AWS 全体のキーを使用しています。

> **Note**
>
> 条件で指定されている `IPAddress` と `NotIpAddress` のキー値は、RFC 4632 の CIDR 表記を使用していることに注意してください。詳細については、http://www.rfc-editor.org/rfc/rfc4632.txt を参照してください。

```
{
    "Version": "2012-10-17",
    "Id": "S3PolicyId1",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": "*",
            "Action":"s3:GetObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
```

```
            "Condition" : {
                "IpAddress" : {
                    "aws:SourceIp": "192.0.2.0/24"
                },
                "NotIpAddress" : {
                    "aws:SourceIp": "192.0.2.188/32"
                }
            }
        }
    ]
}
```

Amazon S3 のポリシーでは、他の AWS 全体の条件キーを使用することもできます。例えば、VPC エンドポイントのバケットポリシーで aws:SourceVpce 条件キーや aws:SourceVpc 条件キーを指定できます。具体的な例については、「バケットポリシーを使用した VPC エンドポイントからのアクセスコントロール」を参照してください。

> **Note**
>
> 一部の AWS グローバル条件キーでは、特定のリソースタイプのみがサポートされます。そのため、使用するグローバル条件キーとリソースタイプを Amazon S3 がサポートしているかどうか、または代わりに Amazon S3 固有の条件キーを使用する必要があるかどうかを確認してください。Amazon S3 に対してサポートされているリソースタイプと条件キーの完全なリストについては、サービス認証リファレンスの「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

例 2: 特定のプレフィックスを持つバケット内のオブジェクト一覧の取得

s3:prefix 条件キーを使用して、GET Bucket (ListObjects) API の応答を、特定のプレフィックスを持つキー名に制限できます。バケット所有者であれば、バケット内の特定のプレフィックスの内容に限り、コンテンツを表示することをユーザーに許可できます。この条件キーは、バケットのオブジェクトがキー名プレフィックスによって整理されている場合に便利です。Amazon S3 コンソールでは、キー名のプレフィックスを使用して、フォルダの概念を示します。フォルダの概念をサポートしているはコンソールのみです。Amazon S3 API では、バケットとオブジェクトのみがサポートされています。プレフィックスと区切り文字を使用してアクセス許可をフィルタリングする方法の詳細については、ユーザーポリシーを使用したバケットへのアクセスの制御 を参照してください。

例えば、キー名が public/object1.jpg および public/object2.jpg である 2 つのオブジェクトがある場合、コンソールには public フォルダ以下のオブジェクトが表示されます。Amazon S3

API では、これらはプレフィックスを持つオブジェクトとして扱われ、フォルダ内のオブジェクトとしては扱われません。ただし、Amazon S3 API でこのようなプレフィックスを使用してオブジェクトのキーを管理している場合は、s3:ListBucket のアクセス許可の条件に s3:prefix を付けて付与することで、この特定のプレフィックスの付いたキー名の一覧を取得することができます。

この例では、バケット所有者とユーザーが属する親アカウントは同一です。したがって、バケット所有者はバケットポリシーまたはユーザーポリシーのどちらでも使用することができます。GET Bucket (ListObjects) API で使用できるその他の条件キーの詳細については、ListObjects を参照してください。

ユーザーポリシー

以下のユーザーポリシーは、ユーザーがリクエストで s3:ListBucket およびその値に を指定することを条件として、 のアクセス許可 (prefixバケットの GET (ListObjects)projects を参照) を付与します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:ListBucket",
         "Resource":"arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringNotEquals" : {
                 "s3:prefix": "projects"
             }
          }
       }
    ]
```

```
}
```

この条件では、ユーザーの取得できるリストが projects プレフィックスの付いているオブジェクトキーに限定されます。明示的な拒否を追加すると、ユーザーに付与されている他のアクセス許可に関係なしに、他のプレフィックスが付いたキーのリストを求めるユーザーのリクエストは拒否されます。例えば、以前のユーザーポリシーの更新やバケットポリシーにより、オブジェクトキーのリストを表示するアクセス許可がユーザーに制限なく付与される場合があります。明示的な拒否が常に優先されるため、プレフィックスが projects 以外のキーの表示を求めるユーザーのリクエストは拒否されます。

バケットポリシー

上記のユーザーポリシーに Principal エレメントを追加して、ユーザーを指定する場合は、次のようにバケットポリシーを使用できます。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/bucket-owner"
         },
         "Action":  "s3:ListBucket",
         "Resource": "arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/bucket-owner"
         },
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringNotEquals" : {
```

```
                "s3:prefix": "projects"
            }
        }
    }
  ]
}
```

AWS CLI でポリシーをテストする

このポリシーは、次の `list-object` AWS CLI コマンドを使用してテストできます。このコマンドでは、`--profile` パラメータを使用してユーザーの認証情報を指定します。AWS CLI のセットアップおよび使用の詳細については、AWS CLI を使用した Amazon S3 での開発 を参照してください。

```
aws s3api list-objects --bucket awsexamplebucket1 --prefix examplefolder --profile
 AccountADave
```

バケットがバージョニングに対応している場合、バケット内のオブジェクトのリストを表示するには、`s3:ListBucket` のアクセス許可ではなく、前述のポリシーの `s3:ListBucketVersions` のアクセス許可を付与する必要があります。このアクセス許可は、`s3:prefix` 条件キーもサポートしています。

例 3: キーの最大数の設定

`s3:max-keys` 条件キーを使用すると、リクエスタがバケットの GET (ListObjects) または ListObjectVersions のリクエストで返すことができるキーの最大数を設定できます。デフォルトでは、API は最大 1,000 個のキーを返します。`s3:max-keys` で使用できる数値条件演算子の一覧とその例については、IAM ユーザーガイドの数値条件演算子 を参照してください。

Amazon S3 のアイデンティティベースのポリシー

デフォルトでは、ユーザーとロールには Amazon S3 リソースを作成または変更するアクセス許可がありません。また、AWS Management Console、AWS Command Line Interface (AWS CLI)、または AWS API を使用してタスクを実行することもできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、IAM ユーザーガイドのIAM ポリシーの作成を参照してください。

Amazon S3 が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、条件キー」を参照してください。

トピック

- ポリシーのベストプラクティス
- Amazon S3 のアイデンティティベースのポリシー例
- ユーザーポリシーを使用したバケットへのアクセスの制御

ポリシーのベストプラクティス

アイデンティティベースのポリシーは、あるユーザーがアカウントの Amazon S3 リソースを作成、アクセス、削除できるどうかを決定します。これらのアクションを実行すると、AWS アカウント に料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS マネージドポリシーを使用して開始し、最小特権の許可に移行する – ユーザーとワークロードへの許可の付与を開始するには、多くの一般的なユースケースのために許可を付与する AWS マネージドポリシーを使用します。これらは AWS アカウントで使用できます。ユースケースに応じた AWS カスタマー管理ポリシーを定義することで、許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWSマネージドポリシー」または「ジョブ機能の AWS マネージドポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、IAM ユーザーガイドのIAM でのポリシーとアクセス許可を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。また、AWS CloudFormation などの特定の AWS のサービス を介して使用する場合、条件を使用してサービスアクションへのアクセスを許可することもできます。詳細については、「IAM ユーザーガイド」の [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素:条件) を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは

100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、IAM ユーザーガイドのIAM Access Analyzer ポリシーの検証を参照してください。

- 多要素認証 (MFA) を要求する – AWS アカウント で IAM ユーザーまたはルートユーザーを要求するシナリオがある場合は、セキュリティを強化するために MFA をオンにします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイドのMFA 保護 API アクセスの設定を参照してください。

IAM でのベストプラクティスの詳細については、IAM ユーザーガイドのIAM でのセキュリティのベストプラクティスを参照してください。

Amazon S3 のアイデンティティベースのポリシー例

このセクションでは、Amazon S3 へのユーザーアクセスを管理するための AWS Identity and Access Management (IAM) アイデンティティベースポリシーをいくつか示します。バケットポリシー (リソースベースのポリシー) の例については、「Amazon S3 のバケットポリシー」を参照してください。IAM ポリシー言語については、「Amazon S3 のポリシーとアクセス許可」を参照してください。

次のサンプルポリシーは、プログラムで使用する場合に機能します。ただし、Amazon S3 コンソールでこれらを使用するには、コンソールに必要な追加のアクセス許可を付与する必要があります。このようなポリシーの Amazon S3 コンソールでの使用の詳細については、ユーザーポリシーを使用したバケットへのアクセスの制御 を参照してください。

トピック

- バケット の 1 つへのアクセスを IAM ユーザーに許可する
- バケット内のフォルダへのアクセスを各 IAM ユーザーに許可する
- Amazon S3 で共有フォルダを持つことをグループに許可する
- すべてのユーザーに対し、バケットの特定部分のオブジェクトの読み取りを許可する
- パートナーに対し、バケットの特定部分へのファイルのドロップを許可する
- 特定の AWS アカウント の Amazon S3 バケットへのアクセスを制限する
- 組織単位内の Amazon S3 バケットへのアクセスを制限する
- 組織内の Amazon S3 バケットへのアクセスを制限する
- AWS アカウント に PublicAccessBlock 設定を取得するアクセス許可を付与する
- バケット作成を 1 つのリージョンに制限する

バケット の 1 つへのアクセスを IAM ユーザーに許可する

この例では、AWS アカウント の IAM ユーザーに *example-s3-bucket1* という 1 つのバケットに対する許可を付与して、ユーザーがオブジェクトを追加、更新、削除できるようにします。

このポリシーでは、ユーザーに s3:PutObject、s3:GetObject、s3:DeleteObject のアクセス許可を付与するだけでなく、s3:ListAllMyBuckets、s3:GetBucketLocation、および s3:ListBucket のアクセス許可も付与します。これらが、コンソールで必要とされる追加のアクセス許可です。またコンソール内のオブジェクトのコピー、カット、貼り付けを行うためには、s3:PutObjectAcl および s3:GetObjectAcl アクションが必要となります。コンソールを使用してユーザーにアクセス許可を付与してテストする例の解説については、ユーザーポリシーを使用したバケットへのアクセスの制御 を参照してください。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": "s3:ListAllMyBuckets",
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":["s3:ListBucket","s3:GetBucketLocation"],
         "Resource":"arn:aws:s3:::example-s3-bucket1"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:GetObject",
            "s3:GetObjectAcl",
            "s3:DeleteObject"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/*"
      }
   ]
}
```

バケット内のフォルダへのアクセスを各 IAM ユーザーに許可する

次の例では、Mary と Carlos という 2 人の IAM ユーザーに *example-s3-bucket1* というバケットに対するアクセス許可を付与して、この 2 人がオブジェクトを追加、更新、削除できるようにします。ただし、バケット内の単一のプレフィックス (フォルダ) に全ユーザーのアクセスを制限したいとします。フォルダを作成する際、ユーザー名と同じフォルダ名にすることもできます。

```
example-s3-bucket1
   Mary/
   Carlos/
```

各ユーザーに本人のフォルダのみへのアクセス権を付与するには、各ユーザー用のポリシーを作成し、個別にアタッチします。例えば、Mary に次のポリシーをアタッチして、*example-s3-bucket1*/*Mary* フォルダに対する専用の Amazon S3 のアクセス許可を付与することができます。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/Mary/*"
      }
   ]
}
```

その後、ユーザー Carlos に同様のポリシーをアタッチし、Resource 値のフォルダ *Carlos* を指定します。

各ユーザーにポリシーをアタッチするのではなく、ポリシー変数を使用する単一のポリシーを作成し、そのポリシーをグループにアタッチできます。まずグループを作成し、Mary と Carlos をいずれもそのグループに追加する必要があります。次のポリシーの例では、*example-s3-bucket1*/${aws:username} フォルダに対する Amazon S3 の一連のアクセス許可を付与しています。ポリシーが評価されると、ポリシー変数 ${aws:username} はリクエスタのユーザー名で置き換えられます。例えば、Mary がオブジェクトの PUT リクエストを送信した場合、Mary が *example-s3-*

bucket1/Mary フォルダにオブジェクトをアップロードする PUT オペレーションのみが許可されます。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/${aws:username}/*"
      }
   ]
}
```

> **Note**
>
> ポリシー変数を使用する場合は、2012-10-17 をポリシー内で明示的に指定する必要があります。IAM ポリシー言語のデフォルトバージョンは 2008-10-17 です。このバージョンでは、ポリシー変数をサポートしていません。

Amazon S3 コンソールで前述のポリシーをテストする場合、次のポリシーに示すように、追加のアクセス許可が必要となります。これらのアクセス許可をコンソールで使用する方法については、ユーザーポリシーを使用したバケットへのアクセスの制御 を参照してください。

```
{
 "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGroupToSeeBucketListInTheConsole",
      "Action": [
       "s3:ListAllMyBuckets",
       "s3:GetBucketLocation"
      ],
      "Effect": "Allow",
```

```
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "AllowRootLevelListingOfTheBucket",
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::example-s3-bucket1",
      "Condition":{
            "StringEquals":{
                   "s3:prefix":[""], "s3:delimiter":["/"]
                         }
               }
    },
    {
      "Sid": "AllowListBucketOfASpecificUserPrefix",
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::example-s3-bucket1",
      "Condition":{  "StringLike":{"s3:prefix":["${aws:username}/*"] }
       }
    },
      {
     "Sid": "AllowUserSpecificActionsOnlyInTheSpecificUserPrefix",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/${aws:username}/*"
      }
  ]
}
```

> **Note**
>
> 2012-10-17 バージョンのポリシーでは、ポリシー変数の先頭には $ が付きます。使用するオブジェクトキー (オブジェクト名) に $ が含まれている場合、この構文の変化により衝突が発生します。

この競合を回避するには、$ を使用して ${$} 文字を指定します。例えば、ポリシーにオブジェクトキー my$file を含めるには、my${$}file として指定します。

IAM ユーザー名は人間が読んで理解できるわかりやすい識別子ですが、グローバルで一意である必要はありません。例えば、Carlos が退職して別の Carlos が入社した場合、この別の Carlos が前の Carlos の情報にアクセスできます。

フォルダを作成する際に、ユーザー名の代わりに IAM ユーザー ID を使用することもできます。IAM ユーザー ID はそれぞれ一意であるためです。この場合、${aws:userid} ポリシー変数を使用するように前述のポリシーを修正する必要があります。ユーザー ID の詳細については、「IAM ユーザーガイド」の「IAM 識別子」を参照してください。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/home/${aws:userid}/*"
      }
   ]
}
```

バケット内のフォルダへのアクセスを非 IAM ユーザー (モバイルアプリユーザー) に許可する

ユーザーのデータを S3 バケットに保存するモバイルゲームアプリを開発するとします。バケットに各アプリユーザーのフォルダを作成します。また、各ユーザーには本人のフォルダのみにアクセスを制限します。 ただし、ユーザーがアプリをダウンロードしてゲームをプレイし始める前にフォルダを作成することはできません。お客様にはユーザーのユーザー ID がないためです。

この場合、ユーザーには、Login with Amazon、 Facebook、または Google などのパブリックアイデンティプロバイダを使用してアプリにサインインするよう要求できます。ユーザーがこれらのプロバイダの 1 つを使用してアプリにサインインすると、ユーザー ID が設定されるため、お客様はこれを使用して実行時にユーザー固有のフォルダを作成することができます。

これにより、AWS Security Token Service のウェブ認証フェデレーションを使用して、アイデンティプロバイダからの情報をアプリに組み入れ、各ユーザーの一時的なセキュリティ認証情報を取得することができます。続いて、IAM ポリシーを作成して、アプリがバケットにアクセスできるようにしたり、ユーザー固有のフォルダの作成、データのアップロードなどのオペレーションを実行できるようにすることができます。ウェブ ID フェデレーションの詳細については、IAM ユーザーガイドのウェブ ID フェデレーションについてを参照してください。

Amazon S3 で共有フォルダを持つことをグループに許可する

次のポリシーをグループにアタッチすることで、グループ内の全メンバーに Amazon S3 のフォルダ *example-s3-bucket1*/share/marketing へのアクセス権が付与されます。グループメンバーは、この指定のフォルダのオブジェクトに対してのみ、ポリシーに示されている Amazon S3 の特定のアクセス許可を付与されます。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/share/marketing/*"
      }
   ]
}
```

すべてのユーザーに対し、バケットの特定部分のオブジェクトの読み取りを許可する

次の例では、*AllUsers* というグループを作成し、AWS アカウントのすべての IAM ユーザーを含めます。次に、GetObject フォルダ内のオブジェクトに対してのみ GetObjectVersion と *example-s3-bucket1*/readonly のアクセス権をグループに付与するポリシーをアタッチします。

```
{
   "Version":"2012-10-17",
   "Statement":[
```

```
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/readonly/*"
      }
   ]
}
```

パートナーに対し、バケットの特定部分へのファイルのドロップを許可する

次の例では、パートナー会社を表す *AnyCompany* というグループを作成します。パートナー会社でアクセス許可が必要な個人やアプリケーションのために IAM ユーザーを作成し、そのユーザーをグループに入れます。

次に、バケット内の次のフォルダに対する PutObject アクセス権をグループに付与するポリシーをアタッチします。

example-s3-bucket1/uploads/anycompany

このバケットに対する他の操作を *AnyCompany* グループに禁止するには、PutObjectで Amazon S3 のリソースに対する AWS アカウント 以外の Amazon S3 のアクションを明示的に拒否するステートメントを追加します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::example-s3-bucket1/uploads/anycompany/*"
      },
      {
         "Effect":"Deny",
         "Action":"s3:*",
         "NotResource":"arn:aws:s3:::example-s3-bucket1/uploads/anycompany/*"
      }
   ]
}
```

特定の AWS アカウント の Amazon S3 バケットへのアクセスを制限する

Amazon S3 プリンシパルが信頼された内のリソースにのみアクセスしていることを確認する場合 AWS アカウントでは、アクセスを制限できます。たとえば、このアイデンティティベースの IAM ポリシーは、Deny 効果を使用して Amazon S3 アクションへのアクセスをブロックします。ただし、アクセスされている Amazon S3 リソースがアカウント *222222222222* 内の場合は除きます。AWS アカウント の IAM プリンシパルがカウント外の Amazon S3 オブジェクトにアクセスすることを防止するには、次の IAM ポリシーをアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyS3AccessOutsideMyBoundary",
      "Effect": "Deny",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "222222222222"
          ]
        }
      }
    }
  ]
}
```

> **Note**
>
> このポリシーは、アクセス権を付与しないため、既存の IAM アクセスコントロールに置き換わるものではありません。代わりに、このポリシーは、他の IAM ポリシーによって付与されたアクセス権限に関係なく、他の IAM アクセス権限の追加ガードレールとして機能します。

ポリシーのアカウント ID *222222222222* を自身のポリシー AWS アカウント に必ず置き換えます。この制限を維持しながら複数のアカウントにポリシーを適用するには、アカウント ID を

aws:PrincipalAccount 条件キーに置き換えます。この条件では、プリンシパルとリソースが同じアカウントにある必要があります。

組織単位内の Amazon S3 バケットへのアクセスを制限する

AWS Organizations でセットアップされた組織単位 (OU)がある場合、Amazon S3 バケットのアクセスを組織の特定部分に制限することができます。この例では、aws:ResourceOrgPaths キーを使って組織の OU へ Amazon S3 バケットのアクセスを制限します。この例では、OU ID は *ou-acroot-exampleou* です。自身のポリシーでこの値を自身の OU ID に置き換えてください。

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Sid": "AllowS3AccessOutsideMyBoundary",
       "Effect": "Allow",
       "Action": [
         "s3:*"
       ],
       "Resource": "*",
       "Condition": {
         "ForAllValues:StringNotLike": {
           "aws:ResourceOrgPaths": [
             "o-acorg/r-acroot/ou-acroot-exampleou/"
           ]
         }
       }
     }
   ]
 }
```

> **Note**
>
> このポリシーは、アクセス許可を付与しません。代わりに、このポリシーは、他の IAM アクセス権限のバックストップとして機能し、プリンシパルが OU 定義の境界外にある Amazon S3 オブジェクトにアクセスするのを防ぎます。

このポリシーは、アクセスされている Amazon S3 オブジェクトが組織内の *ou-acroot-exampleou* OU 内に存在しない場合、Amazon S3 アクションへのアクセスを拒否します。IAM ポリシー条件は、リストされた OU パスを含めるのに、複数値を持つ条件キー

aws:ResourceOrgPaths を要求します。このポリシーは、ForAllValues:StringNotLike 演算子を使用して aws:ResourceOrgPaths の値をリストされた OU と比較します (大文字と小文字は区別しません)。

組織内の Amazon S3 バケットへのアクセスを制限する

組織内の Amazon S3 オブジェクトへのアクセスを制限するには、組織のルートに IAM ポリシーをアタッチし、組織内のすべてのアカウントに適用します。IAM プリンシパルにこのルールに従うように要求するには、サービスコントロールポリシー (SCP) を使用します。SCP を使用する場合は、組織のルートにポリシーをアタッチする前に、SCP のテストをしっかりと実行してください。

次のポリシー例では、アクセスされている Amazon S3 オブジェクトが、それにアクセスしている IAM プリンシパルと同じ組織内に存在しない限り、Amazon S3 アクションへのアクセスが拒否されます。

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Sid": "DenyS3AccessOutsideMyBoundary",
       "Effect": "Deny",
       "Action": [
         "s3:*"
       ],
       "Resource": "arn:aws:s3:::*/*",
       "Condition": {
         "StringNotEquals": {
           "aws:ResourceOrgID": "${aws:PrincipalOrgID}"
         }
       }
     }
   ]
 }
```

> **Note**
>
> このポリシーは、アクセス許可を付与しません。代わりに、このポリシーは、他の IAM アクセス権限のバックストップとして機能し、プリンシパルが組織外にある Amazon S3 オブジェクトにアクセスするのを防ぎます。このポリシーは、ポリシーが有効になった後に作成される Amazon S3 リソースにも適用されます。

この例の IAM ポリシー条件では、互いに等しい aws:ResourceOrgID および aws:PrincipalOrgID が必要です。この要件では、リクエストを行うプリンシパルとアクセスされるリソースが同じ組織内になければなりません。

AWS アカウント に PublicAccessBlock 設定を取得するアクセス許可を付与する

以下のアイデンティティベースのポリシーの例は、ユーザーに s3:GetAccountPublicAccessBlock のアクセス許可を付与します。これらのアクセス許可については、Resource 値を "*" に設定します。リソース ARN については、「Amazon S3 のポリシーリソース」を参照してください。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
       }
    ]
}
```

バケット作成を 1 つのリージョンに制限する

例えば、AWS アカウント の管理者がユーザー (Dave) に南米 (サンパウロ) リージョンでのみバケットを作成できる許可を付与するとします。アカウント管理者は、以下のように条件を指定して、s3:CreateBucket のアクセス許可を付与する次のユーザーポリシーをアタッチします。Condition ブロックのキーと値のペアは、s3:LocationConstraint キーと、その値として sa-east-1 リージョンを指定します。

> **Note**
>
> この例では、バケット所有者はユーザーの 1 人にアクセス許可を付与するため、バケットポリシーまたはユーザーポリシーのどちらでも使用することができます。この例では、ユーザーポリシーを使用します。

Amazon S3 リージョンのリストについては、AWS 全般のリファレンス の「リージョンとエンドポイント」を参照してください。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
         "Condition": {
             "StringLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       }
    ]
}
```

明示的な拒否を追加する

上記のポリシーは、ユーザーが sa-east-1 以外のリージョンでバケットを作成することを制限します。ただし、他の一部のポリシーで、このユーザーに別のリージョンでバケットを作成するアクセス許可を付与する場合があります。例えば、ユーザーがグループに属している場合、グループのアクセス許可内にいるすべてのユーザーが別のリージョンでバケットを作成できるように、ポリシーがアタッチされている場合があります。ユーザーに別のリージョンでバケットを作成するアクセス許可が付与されないように、上記のポリシーに明示的な拒否のステートメントを追加します。

Deny ステートメントは、StringNotLike 条件を使用します。つまり、場所の制約が sa-east-1 でない場合、バケットの作成リクエストは拒否されます。明示的な拒否を使用すれば、どのようなアクセス権限が付与されている場合でも、ユーザーは別のリージョンでバケットを作成できなくなります。次のポリシーには、明示的な拒否ステートメントが含まれています。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
```

```
          "Condition": {
              "StringLike": {
                  "s3:LocationConstraint": "sa-east-1"
              }
          }
        },
       {
          "Sid":"statement2",
          "Effect":"Deny",
          "Action": "s3:CreateBucket",
          "Resource": "arn:aws:s3:::*",
          "Condition": {
              "StringNotLike": {
                  "s3:LocationConstraint": "sa-east-1"
              }
          }
        }
    ]
}
```

AWS CLI でポリシーをテストする

このポリシーは、次の `create-bucket` AWS CLI コマンドを使用してテストできます。この例では、`bucketconfig.txt` ファイルを使用して場所の制約を指定しています。Windows ファイルパスに注目してください。バケットの名前とパスを適切に更新する必要があります。`--profile` パラメータを使用して、ユーザーの認証情報を指定する必要があります。AWS CLI のセットアップおよび使用の詳細については、AWS CLI を使用した Amazon S3 での開発 を参照してください。

```
aws s3api create-bucket --bucket examplebucket --profile AccountADave --create-bucket-
configuration file://c:/Users/someUser/bucketconfig.txt
```

`bucketconfig.txt` ファイルは、次のように設定を指定します。

```
{"LocationConstraint": "sa-east-1"}
```

ユーザーポリシーを使用したバケットへのアクセスの制御

このチュートリアルでは、Amazon S3 でのユーザーアクセス許可の使用について説明します。この例では、フォルダを含むバケットを作成します。AWS アカウントで AWS Identity and Access Management IAM ユーザーを作成し、作成したユーザーに対して、Amazon S3 バケットおよびバケット内のフォルダへのアクセス許可を段階的に付与します。

トピック

バケットとフォルダの基本

Amazon S3 のデータモデルはフラットな構造をしています。バケットを作成し、バケットにオブジェクトを保存します。サブバケットやサブフォルダの階層はありませんが、フォルダ階層を エミュレートすることができます。Amazon S3 コンソールなどのツールを使用すると、バケット内の論理フォルダやサブフォルダを表示できます。

companybucket という名前のバケットには、Private、Development、Finance の 3 つのフォルダと s3-dg.pdf オブジェクトが含まれていることがコンソールに表示されます。コンソールでは、フォルダおよびサブフォルダからなる論理的な階層が作成するために、オブジェクト名 (キー) を使用します。次の例を考えます。

- Development フォルダを作成すると、コンソールによって Development/ というキーを持つオブジェクトが作成されます。区切り記号として末尾のスラッシュ (/) が付いている点に注意してください。
- Projects1.xls というオブジェクトを Development フォルダにアップロードすると、コンソールによってそのオブジェクトがアップロードされ、Development/Projects1.xls というキーが設定されます。

  このキーの Development はプレフィックスで、/ は区切り記号です。Amazon S3 API のオペレーションではプレフィックスと区切り記号がサポートされます。例えば、特定のプレフィック

スと区切り記号を持つすべてのオブジェクトの一覧を取得できます。コンソールで Development フォルダを開くと、そのフォルダ内のオブジェクトがコンソールに一覧表示されます。次の例では、Development フォルダには 1 つのオブジェクトが含まれています。

コンソールで Development バケットの companybucket フォルダを表示すると、プレフィックスに Development と区切り文字に / を指定したリクエストが Amazon S3 に送信されます。コンソールのレスポンスは、コンピュータのファイルシステムのフォルダ一覧と似ています。前述の例は、バケット companybucket に Development/Projects1.xls というキーを持つ 1 つのオブジェクトがあることを示しています。

コンソールは、オブジェクトのキーを使用して論理階層を推測しています。Amazon S3 には物理階層はありません。Amazon S3 にはフラットなファイル構造にオブジェクトを含むバケットがあるだけです。Amazon S3 API を使用してオブジェクトを作成すると、オブジェクトのキーを使用して論理階層を表すことができます。オブジェクトの論理階層を作成する場合は、個別のフォルダへのアクセスを管理できます。このチュートリアルで、その方法について説明します。

開始する前に、ルートレベルのバケットの内容の概念を理解する必要があります。companybucket バケットに次のオブジェクトがあるとします。

- Private/privDoc1.txt
- Private/privDoc2.zip
- Development/project1.xls
- Development/project2.xls
- Finance/Tax2011/document1.pdf
- Finance/Tax2011/document2.pdf
- s3-dg.pdf

これらのオブジェクトキーによって、ルートレベルフォルダ Private、Development、および Finance と、ルートレベルオブジェクト s3-dg.pdf を持つ論理階層が作成されます。Amazon S3 コンソールでバケット名を選択すると、ルートレベルの項目が表示されます。コンソールにはトップレベルのプレフィックス (Private/、Development/、Finance/) が表示されます。オブジェクトキー s3-dg.pdf にはプレフィックスがないため、ルートレベル項目として表示されます。

チュートリアルの概要

このチュートリアルでは、3 つのフォルダ (Private、Development、Finance) を含むバケットを作成します。

ユーザーは Alice と Bob の 2 名です。Alice には Development フォルダにのみ、Bob には Finance フォルダにのみアクセスを付与します。Private フォルダの内容は非公開のままにします。このチュートリアルの例では、IAM ユーザー (ユーザー名 Alice と Bob) を作成してアクセスを管理し、必要なアクセス許可を付与します。

IAM では、ユーザーグループを作成して、そのグループのすべてのユーザーに適用するグループレベルのアクセス許可を付与することもできます。これにより、アクセス許可を効果的に管理できます。この演習では、Alice と Bob に共通のアクセス許可が必要です。そのため、Consultants というグループを作成し、Alice と Bob をそのグループに追加します。最初に、グループポリシーをグループに関連付けてアクセス許可を付与します。次に、特定のユーザーにポリシーを関連付けてユーザー固有のアクセス許可を追加します。

> **Note**
>
> このチュートリアルでは、バケット名に companybucket、IAM ユーザーに Alice と Bob、グループ名に Consultants を使用しています。Amazon S3 ではバケット名はグローバルに一意である必要があるため、このバケット名は実際に作成する名前に置き換えてください。

チュートリアルの準備をする

この例では、AWS アカウント 認証情報を使用して、IAM ユーザーを作成します。最初、これらのユーザーにはアクセス許可が何もありません。これらのユーザーに、Amazon S3 の特定のアクションを実行するためのアクセス許可を段階的に付与します。アクセス許可をテストするには、各ユーザーの認証情報を使用してコンソールにサインインします。AWS アカウントの所有者として許可を段階的に付与し、IAM ユーザーとして許可をテストするには、そのたびに異なる認証情報を使用してサインイン/サインアウトする必要があります。これは、1 つのブラウザでテストできますが、2 つのブラウザを使用した方が迅速に処理することができます。一方のブラウザで自分の AWS アカウント 認証情報を使用して AWS Management Console に接続し、もう一方のブラウザで IAM ユーザーの認証情報を使用して接続します。

AWS アカウント 認証情報を使用して AWS Management Console にサインインするには、https://conso4le.aws.amazon.com/ にアクセスします。。IAM ユーザーは、このリンクからはサインインで

きません。IAM ユーザーは IAM 対応のサインインページを使用する必要があります。アカウント所有者からユーザーにこのリンクを通知します。

IAM の詳細については、IAM ユーザーガイドAWS Management Consoleの へのサインインページを参照してください。

IAM ユーザーにサインインのリンクを提供するには

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. [Navigation (ナビゲーション)] ペインで、[IAM Dashboard (IAM ダッシュボード)] を選択します。
3. [IAM users sign in link (IAM ユーザーのサインインのリンク)] の URL を書き留めます。このリンクを IAM ユーザーに提供して、IAM ユーザー名とパスワードを使用してコンソールにサインインしてもらいます。

ステップ 1: バケットを作成する

このステップでは、AWS アカウント認証情報を使用して Amazon S3 コンソールにサインインし、バケットの作成、バケットへのフォルダの追加、各フォルダのへの 1 つまたは 2 つのサンプルドキュメントのアップロードを行います。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. バケットを作成します。

   手順については、「バケットの作成」を参照してください。
3. バケットにドキュメントを 1 つアップロードします。

   この演習では、このバケットのルートレベルに `s3-dg.pdf` というドキュメントがあることを前提としています。別のドキュメントをアップロードする場合は、`s3-dg.pdf` をそのドキュメントのファイル名に置き換えてください。
4. 3 つのフォルダ (`Private`、`Finance`、`Development`) をバケットに追加します。

   フォルダを作成するステップバイステップの手順については、Amazon Simple Storage Service ユーザーガイドの フォルダを使用して Amazon S3 コンソールのオブジェクトを整理する を参照してください。
5. 各フォルダにドキュメントを 1 つか 2 つアップロードします。

この演習では、各フォルダに数個のドキュメントをアップロード済みで、バケットに次のキーを持つオブジェクトが存在することを前提としています。

- `Private/privDoc1.txt`
- `Private/privDoc2.zip`
- `Development/project1.xls`
- `Development/project2.xls`
- `Finance/Tax2011/document1.pdf`
- `Finance/Tax2011/document2.pdf`
- `s3-dg.pdf`

手順については、「オブジェクトのアップロード」を参照してください。

ステップ 2: IAM ユーザーとグループを作成する

IAM コンソールを使用して 2 人の IAM ユーザー (Alice と Bob) を AWS アカウント に追加します。詳細な手順については、「IAM ユーザーガイド」の「AWS アカウント での IAM ユーザーの作成」を参照してください。

また、`Consultants` という名前の管理グループを作成します。続いて、ユーザーをこのグループに追加します。詳細な手順については、「IAM ユーザーグループの作成」を参照してください。

> **Warning**
>
> ユーザーとグループを追加するときに、これらのユーザーにアクセス許可を付与するポリシーを関連付けないでください。最初の時点では、アクセス許可はこれらのユーザーに付与せず、以降のセクションで段階的に付与します。まず、これらの IAM ユーザーにパスワードを割り当てておく必要があります。それらのユーザー認証情報を使用して Amazon S3 のアクションをテストし、アクセス許可が意図したとおりに機能することを確認します。

新しい IAM ユーザーの作成の各手順については、「IAM ユーザーガイド」の「AWS アカウント での IAM ユーザーの作成」を参照してください。このチュートリアル用にユーザーを作成するときは、[AWS Management Console アクセス] を選択し、[プログラムによるアクセス] を消去します。

管理グループを作成する手順については、IAM ユーザーガイドの最初の IAM 管理者のユーザーおよびグループの作成を参照してください。

ステップ 3: IAM ユーザーにアクセス許可が付与されていないことを確認する

ブラウザを 2 つ使用する場合は、ここで 2 つ目のブラウザからいずれかの IAM ユーザー認証情報を使用して、コンソールにサインインできます。

1. IAM ユーザーのサインインのリンク (「IAM ユーザーにサインインのリンクを提供するには」を参照) から、いずれかの IAM ユーザー認証情報を使用して AWS Management Consoleにサインインします。
2. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。

   アクセスが拒否されることを示すコンソールメッセージを確認します。

次に、アクセス許可を段階的にユーザーに付与します。最初に、両方のユーザーに必要なアクセス許可を付与するグループポリシーをアタッチします。

ステップ 4: グループレベルのアクセス許可を付与する

ユーザーが次の内容を実行できるようにします。

- 親アカウントが所有するすべてのバケットを表示します。これを行うには、Bob と Alice に `s3:ListAllMyBuckets` アクションのためのアクセス許可が必要です。
- `companybucket` バケット内のルートレベルの項目、フォルダ、およびオブジェクトを一覧表示します。これを行うには、Bob と Alice に `s3:ListBucket` バケットに対する `companybucket` アクションのためのアクセス許可が必要です。

次に、これらのアクセス許可を付与するポリシーを作成し、`Consultants` グループにアタッチします。

ステップ 4.1: すべてのバケットのリストを表示するアクセス許可を付与する

このステップでは、親アカウントが所有するすべてのバケットを表示するのに必要な最小限のアクセス許可をユーザーに付与する管理ポリシーを作成します。そのポリシーを `Consultants` グループにアタッチします。その管理ポリシーをユーザーまたはグループにアタッチしたら、親 AWS アカウントが所有するバケットのリストを取得する許可をそのユーザーまたはグループに付与します。

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

   > **Note**
   >
   > ユーザーに許可を付与するため、IAM ユーザーではなく、AWS アカウントの認証情報を使用してサインインします。

2. 管理ポリシーを作成します。

   a. 左側のナビゲーションペインで [ポリシー] を選択し、続いて [ポリシーの作成] を選択します。

   b. [JSON] タブを選択します。

   c. 以下のアクセスポリシーをコピーし、 ポリシーテキストフィールドに貼り付けます。

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "AllowGroupToSeeBucketListInTheConsole",
         "Action": ["s3:ListAllMyBuckets"],
         "Effect": "Allow",
         "Resource": ["arn:aws:s3:::*"]
       }
     ]
   }
   ```

   ポリシーは JSON ドキュメントです。そのドキュメントの `Statement` はオブジェクトの配列であり、各オブジェクトが名前と値のペアを使用してアクセス許可を定義しています。前述のポリシーは、1 つの特定のアクセス許可を定義しています。`Action` はアクセスの種類を指定します。ポリシーの `s3:ListAllMyBuckets` は、定義済みの Amazon S3 アクションです。このアクションは Amazon S3 GET サービスオペレーションをターゲットとし、認証された送信者が所有するすべてのバケットの一覧を返します。`Effect` エレメントの値は、特定のアクセスを許可するかどうかを決定します。

   d. [ポリシーの確認] を選択します。次のページで、[`AllowGroupToSeeBucketListInTheConsole`Name (名前)] フィールドに「」と入力し、[Create policy (ポリシーの作成)] を選択します。

> **Note**
>
> [Summary (概要)] エントリに、このポリシーではどのアクセス許可も付与されないことを示すメッセージが表示されます。このチュートリアルでは、このメッセージを無視しても問題ありません。

3. 作成した `AllowGroupToSeeBucketListInTheConsole` 管理ポリシーを `Consultants` グループにアタッチします。

   管理ポリシーのアタッチの各手順については、「IAM ユーザーガイド」の「IAM ID のアクセス許可の追加および削除」を参照してください。

   IAM コンソールで、ポリシードキュメントを IAM ユーザーおよびグループにアタッチします。両方のユーザーがバケットのリストを取得できるように、ポリシーをグループにアタッチします。

4. アクセス許可をテストします。

   a. IAM ユーザーのサインインのリンク (「IAM ユーザーにサインインのリンクを提供するには」を参照) から、いずれかの IAM ユーザー認証情報を使用してコンソールにサインインします。

   b. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。

      コンソールにすべてのバケットが表示されますが、バケット内のオブジェクトは表示されません。

ステップ 4.2: バケットのルートレベルの内容をユーザーが表示できるようにする

次に、`Consultants` グループのすべてのユーザーがルートレベルの `companybucket` バケット項目を一覧表示できるようにします。Amazon S3 コンソールで会社のバケットを選択すると、そのバケットのルートレベルのアイテムが表示されます。

> **Note**
>
> この例では、`companybucket` を使用して説明します。作成したバケットの名前を使用する必要があります。

バケット名を選択したときにコンソールから Amazon S3 に送信されるリクエスト、Amazon S3 から返されるレスポンス、コンソールでレスポンスがどのように解釈されるかについて理解するために、さらに詳しくフローを見てみましょう。

バケット名を選択すると、コンソールから Amazon S3 にバケットの GET (ListObjects) のリクエストが送信されます。このリクエストには次のパラメータが含まれます。

- 空の文字列を値に持つ `prefix` パラメータ。
- `delimiter` を値に持つ `/` パラメータ。

リクエストの例を次に示します。

```
GET ?prefix=&delimiter=/ HTTP/1.1
Host: companybucket.s3.amazonaws.com
Date: Wed, 01 Aug  2012 12:00:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Amazon S3 から返されるレスポンスには、次の `<ListBucketResult/>` エレメントが含まれます。

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>companybucket</Name>
  <Prefix></Prefix>
  <Delimiter>/</Delimiter>
   ...
  <Contents>
    <Key>s3-dg.pdf</Key>
    ...
  </Contents>
  <CommonPrefixes>
    <Prefix>Development/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>Finance/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>Private/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

キーの s3-dg.pdf オブジェクトには区切り文字のスラッシュ (/) が含まれておらず、このキーは Amazon S3 から <Contents> エレメントで返されています。一方、このバケットの例の他のキーには区切り文字の / が含まれています。Amazon S3 は、キーをグループ化し、<CommonPrefixes>、Development/、および Finance/ のそれぞれのプレフィックス値の Private/ エレメントを返します。このエレメントは、これらのキーの先頭から、指定した / 区切り記号の出現箇所までのサブ文字列です。

コンソールでこの結果が解釈され、ルートレベルの項目が 3 つのフォルダと 1 つのオブジェクトキーとして表示されます。

Bob または Alice が Development フォルダを開くと、コンソールから Amazon S3 にバケットの GET (ListObjects) のリクエストが送信されます。このリクエストの prefix パラメータと delimiter パラメータは、次の値に設定されます。

- 値が prefix の Development/ パラメータ。
- 値が「delimiter」の / パラメータ。

レスポンスでは、指定したプレフィックスで始まるオブジェクトのキーが Amazon S3 から返されます。

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>companybucket</Name>
  <Prefix>Development</Prefix>
  <Delimiter>/</Delimiter>
   ...
  <Contents>
    <Key>Project1.xls</Key>
    ...
  </Contents>
  <Contents>
    <Key>Project2.xls</Key>
    ...
  </Contents>
</ListBucketResult>
```

コンソールにオブジェクトキーが表示されます。

ここで、ルートレベルのバケット項目を一覧表示するためのアクセス許可をユーザーに付与する作業に戻ります。バケットの内容を一覧表示するには、ユーザーは s3:ListBucket アクションを呼び出すためのアクセス許可を持っている必要があります。次のポリシーステートメント を参照して

ください。ルートレベルのコンテンツのみを表示するようにするために、ユーザーがリクエストで prefix に値を指定していないことを条件に追加します。これにより、ユーザーはルートレベルのフォルダをダブルクリックできなくなります。最後に、値が「delimiter」の / パラメータをユーザーリクエストに組み込むようユーザーに義務付けて、フォルダ形式のアクセスを要求する条件を追加します。

```
{
  "Sid": "AllowRootLevelListingOfCompanyBucket",
  "Action": ["s3:ListBucket"],
  "Effect": "Allow",
  "Resource": ["arn:aws:s3:::companybucket"],
  "Condition":{
         "StringEquals":{
             "s3:prefix":[""], "s3:delimiter":["/"]
                       }
          }
}
```

Amazon S3 コンソールでバケットを選択すると、コンソールはまず Get Bucket location リクエストを送信して、バケットがデプロイされている AWS リージョン を確認します。次に、コンソールはリージョン固有のバケットのエンドポイントを使用して、バケットの GET (ListObjects) のリクエストを送信します。結果として、ユーザーがコンソールを使用する場合は、次のポリシーステートメントに示すように、s3:GetBucketLocation アクションのためのアクセス許可を付与する必要があります。

```
{
   "Sid": "RequiredByS3Console",
   "Action": ["s3:GetBucketLocation"],
   "Effect": "Allow",
   "Resource": ["arn:aws:s3:::*"]
}
```

ユーザーがルートレベルのバケットの内容を一覧表示できるようにするには

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

   IAM ユーザーの認証情報ではなく、AWS アカウントの認証情報を使用してコンソールにサインインします。

2. Consultants グループにアタッチされている既存の AllowGroupToSeeBucketListInTheConsole 管理ポリシーを以下のポリシーに置き換えます。このポリシーでも s3:ListBucket アクションは実行できます。Resource ポリシーの *companybucket* は、自分のバケットの名前に置き換えてください。

   手順については、「IAM ユーザーガイド」の「IAM ポリシーの編集」を参照してください。詳細な手順では、ポリシーがアタッチされているすべてのプリンシパルエンティティに変更を適用するステップに従ってください。

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Sid":
 "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",
       "Action": [ "s3:ListAllMyBuckets", "s3:GetBucketLocation" ],
       "Effect": "Allow",
       "Resource": [ "arn:aws:s3:::*"  ]
     },
     {
       "Sid": "AllowRootLevelListingOfCompanyBucket",
       "Action": ["s3:ListBucket"],
       "Effect": "Allow",
       "Resource": ["arn:aws:s3:::companybucket"],
       "Condition":{
             "StringEquals":{
                    "s3:prefix":[""], "s3:delimiter":["/"]
                           }
                  }
     }
  ]
}
```

3. 更新されたアクセス許可をテストします。

   a. IAM ユーザーのサインインリンク (「IAM ユーザーにサインインのリンクを提供するには」を参照) を使用して AWS Management Consoleにサインインします。

      https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。

b. 作成したバケットを選択すると、ルートレベルのバケット項目がコンソールに表示されます。バケット内のフォルダを選択しても、フォルダの内容は表示されません。そのためのアクセス許可が付与されていないからです。

このテストは、ユーザーが Amazon S3 コンソールを使用すると成功します。コンソールでバケットを選択すると、コンソールの実装により、空の文字列を値に持つ `prefix` パラメータと、`delimiter` を値に持つ `/` パラメータが送信されるためです。

ステップ 4.3: グループポリシーの概要

追加したグループポリシーによって、IAM ユーザーである Alice と Bob に次の最小限のアクセス許可が付与されます。

- 親アカウントが所有するすべてのバケットを表示します。
- `companybucket` バケット内のルートレベルの項目を表示します。

ただし、ユーザーが実行できる操作はまだ十分ではありません。次のように、ユーザー固有のアクセス許可を付与します。

- `Development` フォルダのオブジェクトの読み書きを Alice に許可します。
- `Finance` フォルダのオブジェクトの読み書きを Alice と Bob に許可します。

ユーザー固有のアクセスを許可するには、グループではなく特定のユーザーにポリシーを関連付けます。次のセクションでは、`Development` フォルダを操作するためのアクセス許可を Alice に付与します。`Finance` フォルダを操作するための同様のアクセス許可を Bob に付与するステップを繰り返します。

ステップ 5: IAM ユーザーの Alice に特定のアクセス許可を付与する

次に、`Development` フォルダの内容を表示し、そのフォルダ内のオブジェクトを読み書きできるように、追加のアクセス許可を Alice に付与します。

ステップ 5.1: IAM ユーザーの Alice に Development フォルダの内容を一覧表示するためのアクセス許可を付与する

Alice が `Development` フォルダの内容を表示できるようにするには、`s3:ListBucket` バケットに対して `companybucket` アクションを実行するためのアクセス許可を付与するポリシーをユーザー Alice に適用する必要があります (プレフィックス `Development/` がリクエストに含まれていること

を前提とします)。このポリシーはユーザー (Alice) にのみ適用するため、インラインポリシーを使用します。インラインポリシーの詳細については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシー」を参照してください。

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

   IAM ユーザーの認証情報ではなく、AWS アカウントの認証情報を使用してコンソールにサインインします。

2. Development フォルダの内容を表示するアクセス許可をユーザー (Alice) に付与するインラインポリシーを作成します。

   a. 左側のナビゲーションペインで、[ユーザー] を選択します。

   b. ユーザー名 [Alice] を選択します。

   c. ユーザーの詳細ページで、[Permissions (アクセス許可)] タブを選択し、[Add inline policy (インラインポリシーの追加)] を選択します。

   d. [JSON] タブを選択します。

   e. 以下のアクセスポリシーをコピーし、ポリシーのテキストフィールドに貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
      "Sid": "AllowListBucketIfSpecificPrefixIsIncludedInRequest",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::companybucket"],
      "Condition":{  "StringLike":{"s3:prefix":["Development/*"] }
       }
    }
  ]
}
```

   f. [ポリシーの確認] を選択します。次のページで、[Name (名前)] フィールドに名前を入力し、[Create policy (ポリシーの作成)] を選択します。

3. Alice のアクセス許可の変更をテストします。

   a. IAM ユーザーのサインインリンク (「IAM ユーザーにサインインのリンクを提供するには」を参照) を使用して AWS Management Consoleにサインインします。

b. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。

c. Amazon S3 コンソールで、Alice がバケットの Development/ フォルダ内のオブジェクトを一覧表示できることを確認します。

ユーザーが /Development フォルダ内のオブジェクトを一覧表示するためにそのフォルダを選択すると、Amazon S3 コンソールから Amazon S3 にプレフィックスに ListObjects を指定した /Development のリクエストが送信されます。ユーザーにはプレフィックスに Development と区切り文字に / を指定してオブジェクトを一覧表示するアクセス許可が付与されているため、Amazon S3 からキープレフィックスが Development/ のオブジェクトの一覧が返されてコンソールに表示されます。

ステップ 5.2: IAM ユーザーの Alice に Development フォルダでオブジェクトを取得および作成するためのアクセス許可を付与する

Alice が Development フォルダ内のオブジェクトを読み書きできるようにするには、s3:GetObject および s3:PutObject アクションを呼び出すためのアクセス許可が Alice に必要です。次のポリシーステートメントは、そのアクセス許可を付与します (リクエストに prefix の値を持つ Development/ パラメータが設定されているとします)。

```
{
    "Sid":"AllowUserToReadWriteObjectData",
    "Action":["s3:GetObject", "s3:PutObject"],
    "Effect":"Allow",
    "Resource":["arn:aws:s3:::companybucket/Development/*"]
 }
```

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

   IAM ユーザーの認証情報ではなく、AWS アカウントの認証情報を使用してコンソールにサインインします。

2. 前のステップで作成したインラインポリシーを編集します。

   a. 左側のナビゲーションペインで、[ユーザー] を選択します。

   b. ユーザー名 [Alice] を選択します。

   c. ユーザーの詳細で、[アクセス許可] タブを選択し、[インラインポリシー] セクションを展開します。

d. 前のステップで作成したポリシーの名前の横にある [Edit Policy (ポリシーの編集)] をクリックします。

e. 以下のポリシーをコピーして、 ポリシーテキストフィールドに貼り付け、既存のポリシーと置き換えます。

```
{
     "Version": "2012-10-17",
     "Statement":[
      {
         "Sid":"AllowListBucketIfSpecificPrefixIsIncludedInRequest",
         "Action":["s3:ListBucket"],
         "Effect":"Allow",
         "Resource":["arn:aws:s3:::companybucket"],
         "Condition":{
            "StringLike":{"s3:prefix":["Development/*"]
            }
         }
      },
      {
        "Sid":"AllowUserToReadWriteObjectDataInDevelopmentFolder",
        "Action":["s3:GetObject", "s3:PutObject"],
        "Effect":"Allow",
        "Resource":["arn:aws:s3:::companybucket/Development/*"]
      }
   ]
}
```

3. 更新されたポリシーをテストします。

a. IAM ユーザーのサインインリンク (「IAM ユーザーにサインインのリンクを提供するには」を参照) を使用して AWS Management Consoleにサインインします。

b. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。

c. Amazon S3 コンソールで、Alice が Development フォルダに対するオブジェクトの追加およびダウンロードを行えるようになったことを確認します。

ステップ 5.3: バケットの他のフォルダへの IAM ユーザーの Alice のアクセス許可を明示的に拒否する

ユーザー Alice が companybucket バケットのルートレベルの内容を表示できるようになったことを確認します。Alice は Development フォルダのオブジェクトを読み書きすることもできます。厳

密にアクセス許可を制限したい場合は、バケットの他のフォルダへの Alice のアクセスを明示的に拒否することができます。バケットの他のフォルダへのアクセス許可を Alice に付与する他のポリシー (バケットポリシーまたは ACL) が存在する場合は、この明示的な拒否がそれらのアクセス許可よりも優先されます。

次のステートメントをユーザーの Alice のポリシーに追加すると、Alice が Amazon S3 に送信するすべてのリクエストに `prefix` パラメータを含めて値を `Development/*` または空の文字列にすることを要求できます。

```
{
   "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",
   "Action": ["s3:ListBucket"],
   "Effect": "Deny",
   "Resource": ["arn:aws:s3:::companybucket"],
   "Condition":{  "StringNotLike": {"s3:prefix":["Development/*",""] },
                  "Null"         : {"s3:prefix":false }
    }
}
```

`Condition` ブロックには 2 つの条件式があります。これらの条件式の結果は、論理 `AND` を使用して結合されます。両方の条件が true の場合、結合された条件の結果は true です。このポリシーの `Effect` が `Deny` であるため、`Condition` が true と評価されると、ユーザーは指定した `Action` を実行できません。

- `Null` 条件式により、Alice からのリクエストに `prefix` パラメータが含まれていることが保証されます。

  `prefix` パラメータはフォルダに類似したアクセスを必要とします。`prefix` パラメータのないリクエストを送信すると、Amazon S3 によりすべてのオブジェクトキーが返されます。

  リクエストに null 値の `prefix` パラメータが含まれている場合、式の評価結果は true になり、`Condition` 全体の評価結果が true になります。空の文字列を `prefix` パラメータの値として許可する必要があります。null 文字列を許可すると、この前の説明に示したコンソールでの操作と同様に、Alice はルートレベルのバケット項目を取得できるようになります。詳細については、「ステップ 4.2: バケットのルートレベルの内容をユーザーが表示できるようにする」を参照してください。
- 条件式 `StringNotLike` を使用すると、`prefix` パラメータの値が指定されていて `Development/*` でない場合、リクエストは失敗します。

前のセクションの手順に従って、ユーザー Alice 用に作成したインラインポリシーを再び更新します。

以下のポリシーをコピーして、 ポリシーテキストフィールドに貼り付け、既存のポリシーと置き換えます。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllowListBucketIfSpecificPrefixIsIncludedInRequest",
         "Action":["s3:ListBucket"],
         "Effect":"Allow",
         "Resource":["arn:aws:s3:::companybucket"],
         "Condition":{
            "StringLike":{"s3:prefix":["Development/*"]
            }
         }
      },
      {
        "Sid":"AllowUserToReadWriteObjectDataInDevelopmentFolder",
        "Action":["s3:GetObject", "s3:PutObject"],
        "Effect":"Allow",
        "Resource":["arn:aws:s3:::companybucket/Development/*"]
      },
      {
         "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",
         "Action": ["s3:ListBucket"],
         "Effect": "Deny",
         "Resource": ["arn:aws:s3:::companybucket"],
         "Condition":{  "StringNotLike": {"s3:prefix":["Development/*",""] },
                        "Null"         : {"s3:prefix":false }
          }
      }
   ]
}
```

ステップ 6: IAM ユーザーの Bob に特定のアクセス許可を付与する

次に、Finance フォルダへのアクセス許可を Bob に付与します。Alice にアクセス許可を付与するときに使用した手順に従います。ただし、Development フォルダは Finance フォルダに置き換え

ます。手順については、「ステップ 5: IAM ユーザーの Alice に特定のアクセス許可を付与する」を参照してください。

ステップ 7: Private フォルダをセキュリティで保護する

この例では、ユーザーは 2 名だけです。グループレベルで最小限必要なすべてのアクセス許可を付与し、ユーザーレベルのアクセス許可は、個々のユーザーレベルでアクセスを許可することが必要な場合にのみ付与しました。このようにすると、アクセス許可を管理する手間を最小限に抑えることができます。ユーザー数が増えるに従って、アクセス許可の管理は煩雑になります。例えば、この例のどのユーザーも Private フォルダの内容にアクセスできないようにします。Private フォルダへのアクセス許可を誤って付与しないようにするには、どのような方法があるでしょうか。このフォルダへのアクセスを明示的に拒否するポリシーを追加します。明示的な拒否は他のあらゆるアクセス許可よりも優先されます。

Private フォルダを非公開に保つには、次の 2 つの拒否ステートメントをグループポリシーに追加します。

- 次のステートメントを追加して、Private フォルダのリソースに対するあらゆるアクションを明示的に拒否します (companybucket/Private/*)。

```
{
  "Sid": "ExplictDenyAccessToPrivateFolderToEveryoneInTheGroup",
  "Action": ["s3:*"],
  "Effect": "Deny",
  "Resource":["arn:aws:s3:::companybucket/Private/*"]
}
```

- また、リクエストに Private/ プレフィックスが指定されている場合に、オブジェクトを一覧表示するアクションに必要なアクセス許可を拒否します。コンソールで Bob または Alice が Private フォルダを開くと、このポリシーにより Amazon S3 からエラーレスポンスが返されます。

```
{
  "Sid": "DenyListBucketOnPrivateFolder",
  "Action": ["s3:ListBucket"],
  "Effect": "Deny",
  "Resource": ["arn:aws:s3:::*"],
  "Condition":{
      "StringLike":{"s3:prefix":["Private/"]}
   }
}
```

Consultants グループポリシーを、前述の拒否ステートメントを含む更新したポリシーに置き換えます。更新したポリシーが適用されると、グループ内のどのユーザーも、バケット内の Private フォルダにアクセスできなくなります。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

   IAM ユーザーの認証情報ではなく、AWS アカウントの認証情報を使用してコンソールにサインインします。

2. Consultants グループにアタッチされている既存の AllowGroupToSeeBucketListInTheConsole 管理ポリシーを以下のポリシーに置き換えます。ポリシーの *companybucket* は自分のバケットの名前に置き換えてください。

   手順については、「IAM ユーザーガイド」の「カスタマー管理ポリシーの編集 (コンソール)」を参照してください。手順では、ポリシーがアタッチされているすべてのプリンシパルエンティティに変更を適用する指示に従ってください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid":
 "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]
    },
    {
      "Sid": "AllowRootLevelListingOfCompanyBucket",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::companybucket"],
      "Condition":{
          "StringEquals":{"s3:prefix":[""]}
       }
    },
    {
      "Sid": "RequireFolderStyleList",
      "Action": ["s3:ListBucket"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::*"],
```

```
      "Condition":{
          "StringNotEquals":{"s3:delimiter":"/"}
       }
     },
    {
      "Sid": "ExplictDenyAccessToPrivateFolderToEveryoneInTheGroup",
      "Action": ["s3:*"],
      "Effect": "Deny",
      "Resource":["arn:aws:s3:::companybucket/Private/*"]
    },
    {
      "Sid": "DenyListBucketOnPrivateFolder",
      "Action": ["s3:ListBucket"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::*"],
      "Condition":{
          "StringLike":{"s3:prefix":["Private/"]}
       }
    }
  ]
}
```

ステップ 8: クリーンアップする

クリーンアップするには、IAM コンソールを開き、ユーザーの Alice と Bob を削除します。手順については、「IAM ユーザーガイド」の「IAM ユーザーの削除」を参照してください。

ストレージへの不要な請求を防ぐために、この演習で作成したオブジェクトとバケットも削除してください。

関連リソース

- 「IAM ユーザーガイド」の「IAM ポリシーを管理する」

チュートリアル: ポリシーを使用した Amazon S3 リソースへのアクセスの管理

このトピックでは、Amazon S3 リソースへのアクセスの付与について、次の基本的なチュートリアル例を示します。これらの例では、AWS Management Console を使用してリソース (バケット、オブジェクト、ユーザー) を作成し、アクセス許可を付与します。また、これらの例では、コマンドラインツールを使用してアクセス許可を確認する方法を示します。コードを記述する必要はありません。AWS Command Line Interface (AWS CLI) と AWS Tools for Windows PowerShell の両方を使用したコマンドを使用することができます。

- 例 1: バケット所有者がユーザーにバケットのアクセス許可を付与する

  デフォルトでは、使用するアカウントで作成した IAM ユーザーにアクセス許可はありません。この演習では、バケットおよびオブジェクト操作を実行するためのアクセス許可をユーザーに付与します。

- 例 2: バケット所有者がクロスアカウントのバケットのアクセス許可を付与する

  この演習では、バケット所有者 (アカウント A) が別の AWS アカウント (アカウント B) に対してクロスアカウントの許可を付与します。次に、アカウント B でこれらの許可を、そのアカウントのユーザーに委任します。

- オブジェクト所有者とバケット所有者が同じではない場合のオブジェクトのアクセス許可の管理

  このシナリオ例は、バケット所有者が他のユーザーにオブジェクトのアクセス許可を付与しますが、バケット内のすべてのオブジェクトをバケット所有者が所有しているわけではないという場合です。バケット所有者にはどのようなアクセス許可が必要になり、それらのアクセス許可をどのように委任できるのでしょうか。

  バケットを作成する AWS アカウント は、バケット所有者と呼ばれます。この所有者は、オブジェクトをアップロードする許可を他の AWS アカウントに付与することができ、オブジェクトを作成する AWS アカウントがオブジェクトを所有しています。バケット所有者には、他の AWS アカウントで作成されたオブジェクトに対する許可はありません。バケット所有者がオブジェクトへのアクセス権を付与するバケットポリシーを作成する場合、そのポリシーは他のアカウントが所有するオブジェクトには適用されません。

  この場合、オブジェクト所有者は、オブジェクト ACL を使用して、まずバケット所有者にアクセス許可を付与する必要があります。バケット所有者は、以下の例に示すように、これらのオブジェクトの許可を、他のユーザー、自分のアカウントのユーザー、または別の AWS アカウントに委任できます。

  - 例 3: バケット所有者が自分の所有していないオブジェクトに対するアクセス許可を付与する

この演習では、バケット所有者は最初にオブジェクト所有者からアクセス許可を取得します。次に、バケット所有者は、自分のアカウントのユーザーにそのアクセス許可を委任します。

- 例 4 - バケット所有者が所有権のないオブジェクトへのクロスアカウントアクセス許可を付与する

  オブジェクトの所有者からアクセス許可を取得した後も、クロスアカウントの委任がサポートされていないため、バケット所有者は他の AWS アカウント にアクセス許可を委任できません (「アクセス許可の委任」を参照)。代わりに、バケット所有者は、特定のオペレーション (オブジェクトの取得など) を実行する許可を持つ IAM ロールを作成し、別の AWS アカウントがそのロールを引き受けることを許可できます。このロールを引き受けるすべてのユーザーがオブジェクトにアクセスできます。この例では、バケット所有者が IAM ロールを使用してこのクロスアカウントの委任を有効にする方法を示します。

チュートリアル例を実行する前に

これらの例では、AWS Management Console を使用してリソースを作成し、アクセス許可を付与します。許可をテストするために、この例ではコマンドラインツール AWS CLI および AWS Tools for Windows PowerShell を使用するため、コードを書く必要はありません。アクセス許可をテストするには、これらのツールのいずれかをセットアップする必要があります。詳細については、「チュートリアル用のツールのセットアップ」を参照してください。

さらに、これらの例では、リソースの作成時に AWS アカウントのルートユーザー認証情報を使用しません。代わりに、アカウントに、これらのタスクを実行するための管理者ユーザーを作成します。

管理者ユーザーを使用したリソースの作成とアクセス許可の付与について

AWS Identity and Access Management (IAM) では、AWS アカウントのルートユーザー認証情報を使用してリクエストを行わないようお勧めします。代わりに、IAM ユーザーまたはロールを作成してフルアクセスを許可し、そのユーザーまたはロールの認証情報を使用してリクエストを行います。このユーザーを管理者ユーザーまたはロールと呼びます。詳細については、AWS 全般のリファレンスの「AWS アカウントのルートユーザー の認証情報と IAM ID」と IAM ユーザーガイドの「IAM のベストプラクティス」を参照してください。

このセクションのすべてのチュートリアル例では、管理者ユーザーの認証情報を使用します。まだ AWS アカウントの管理者ユーザーを作成していない場合は、トピックでその方法を説明します。

ユーザーの認証情報を使用して AWS Management Console にサインインするには、IAM ユーザーのサインイン URL を使用する必要があります。IAM コンソールは、AWS アカウント にこの URL を提供します。トピックで、この URL を取得する方法を示します。

チュートリアル用のツールのセットアップ

導入例 (チュートリアル: ポリシーを使用した Amazon S3 リソースへのアクセスの管理 を参照) では、AWS Management Console を使用してリソースを作成し、アクセス許可を付与します。許可をテストするために、この例ではコマンドラインツール AWS Command Line Interface (AWS Tools for Windows PowerShell)および AWS CLI を使用するため、コードを書く必要はありません。アクセス許可をテストするには、これらのツールのいずれかをセットアップする必要があります。

AWS CLI をセットアップする方法

1. AWS CLI をダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。

   AWS Command Line Interface の最新バージョンのインストールまたは更新

   AWS Command Line Interface の開始方法
2. デフォルトのプロファイルを設定します。

   ユーザーの認証情報を AWS CLI 設定ファイルに格納します。AWS アカウントの認証情報を使用して設定ファイルにデフォルトのプロファイルを作成します。AWS CLI 設定ファイルの検索と編集の手順については、「Configuration and credential file settings」を参照してください。

   ```
   [default]
   aws_access_key_id = access key ID
   aws_secret_access_key = secret access key
   region = us-west-2
   ```
3. コマンドプロンプトで以下のコマンドを入力して、セットアップを確認します。これらのコマンドは、いずれも認証情報を明示的に提供しないため、デフォルトプロファイルの認証情報が使用されます。

   - help コマンドを試してみます。

     ```
     aws help
     ```
   - 設定したアカウントのバケットのリストを取得するには、aws s3 ls コマンドを使用します。

```
aws s3 ls
```

このチュートリアルでは、以下の例に示すように、ユーザーを作成し、プロファイルを作成することによってユーザー認証情報を設定ファイルに保存します。これらのプロファイルには、AccountAadmin と AccountBadmin の名前があります。

```
[profile AccountAadmin]
aws_access_key_id = User AccountAadmin access key ID
aws_secret_access_key = User AccountAadmin secret access key
region = us-west-2

[profile AccountBadmin]
aws_access_key_id = Account B access key ID
aws_secret_access_key = Account B secret access key
region = us-east-1
```

これらのユーザー認証情報を使用してコマンドを実行するには、プロファイル名を指定する --profile パラメータを追加します。次の AWS CLI コマンドは、*examplebucket* 内のオブジェクトのリストを取得し、AccountBadmin プロファイルを指定します。

```
aws s3 ls s3://examplebucket --profile AccountBadmin
```

また、コマンドプロンプトから AWS_DEFAULT_PROFILE 環境変数を変更することで、ユーザー認証情報セットをデフォルトプロファイルとして設定できます。これを行うと、--profile パラメータを指定せずに AWS CLI コマンドを実行するたびに、AWS CLI は環境変数でデフォルトプロファイルとして設定したプロファイルを使用します。

```
$ export AWS_DEFAULT_PROFILE=AccountAadmin
```

AWS Tools for Windows PowerShell をセットアップする

1. AWS Tools for Windows PowerShell をダウンロードして設定します。手順については、「AWS Tools for Windows PowerShell ユーザーガイド」の「Installing the AWS Tools for Windows PowerShell」を参照してください。

> **Note**
>
> AWS Tools for Windows PowerShell モジュールをロードするには、PowerShell スクリプトの実行を有効にする必要があります。詳細については、「AWS Tools for Windows PowerShell ユーザーガイド」の「Enable Script Execution」を参照してください。

2. これらのチュートリアルでは、`Set-AWSCredentials` コマンドを使用して、セッションごとに AWS 認証情報を指定します。このコマンドは永続的なストア (`-StoreAs` パラメータ) に認証情報を格納します。

```
Set-AWSCredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -
storeas string
```

3. セットアップを確認します。

   - Amazon S3 オペレーションで使用できるコマンドのリストを取得するには、`Get-Command` コマンドを実行します。

```
Get-Command -module awspowershell -noun s3* -StoredCredentials string
```

   - バケット内のオブジェクトのリストを取得するには、`Get-S3Object` コマンドを実行します。

```
Get-S3Object -BucketName bucketname -StoredCredentials string
```

コマンドのリストについては、「AWS Tools for PowerShell Cmdlet Reference」を参照してください。

これで、チュートリアルを行う準備ができました。各セクションの冒頭に示されているリンクをクリックしてください。

例 1: バケット所有者がユーザーにバケットのアクセス許可を付与する

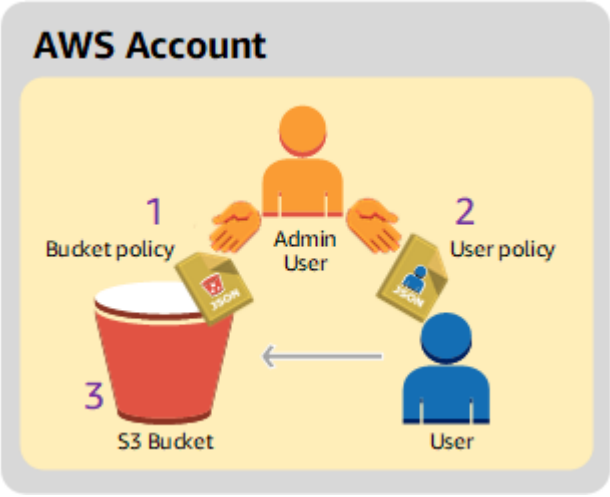
> **Important**
>
> IAM ロールへのアクセス許可の付与は、個別のユーザーへのアクセス許可の付与より優れた方法です。IAM ロールへのアクセス許可の付与方法の詳細については、「クロスアカウントのアクセス許可の理解と IAM ロールの使用」を参照してください。

トピック

- チュートリアルの準備をする
- ステップ 1: アカウント A でリソースを作成して、アクセス許可を付与する
- ステップ 2: アクセス許可をテストする

このチュートリアルでは、AWS アカウント がバケットを所有し、アカウントに IAM ユーザーが含まれています。デフォルトでは、ユーザーにはアクセス許可はありません。何らかのタスクを実行するユーザーには、親アカウントがそのためのアクセス許可をユーザーに付与する必要があります。バケット所有者と親アカウントは同じです。したがって、バケットへの許可をユーザーに付与するには、AWS アカウントは、バケットポリシー、ユーザーポリシー、またはその両方を使用することができます。アカウント所有者は、一部のアクセス許可はバケットポリシーを使用し、他のアクセス許可はユーザーポリシーを使用して付与します。

手順の概要を以下に示します。



1. アカウント管理者は、ユーザーに一連のアクセス許可を付与するバケットポリシーを作成します。

2. アカウント管理者は、ユーザーポリシーをユーザーにアタッチして、追加のアクセス許可を付与します。
3. これによりユーザーは、バケットポリシーとユーザーポリシーの両方を通じて付与されているアクセス許可を使用できます。

この例では、AWS アカウントが必要です。アカウントのルートユーザー認証情報を使用する代わりに、管理者ユーザーを作成します (「管理者ユーザーを使用したリソースの作成とアクセス許可の付与について」を参照してください)。ここ例の AWS アカウント および管理者ユーザーは、次のとおりです。

アカウント ID	アカウントの呼び方	アカウントの管理者ユーザー
1111-1111-1111	アカウント A	AccountAadmin

> **Note**
>
> この例では、管理者ユーザーは AccountAadmin で、アカウント A を参照しており、AccountAdmin ではありません。

ユーザーを作成し、アクセス許可を付与するためのすべてのタスクは、AWS Management Console で実行します。このチュートリアルでは、許可を検証するために、コマンドラインツールの AWS Command Line Interface (AWS CLI) と AWS Tools for Windows PowerShell を使用するので、コードを記述する必要はありません。

チュートリアルの準備をする

1. AWS アカウントが存在し、管理者権限を持つユーザーが存在することを確認します。

   a. 必要な場合、AWS アカウントにサインアップします。このアカウントをアカウント A と呼びます。

      i. https://aws.amazon.com/s3 に移動して、[AWS アカウントの作成] を選択します。

      ii. 画面上の指示に従ってください。

      アカウントがアクティブになり、使用可能な状態になったら、AWS から E メールで通知が届きます。

b. アカウント A に管理者ユーザー **AccountAadmin** を作成します。アカウント A の認証情報を使用して、IAM コンソールにサインインし、次の操作を行います。

i. ユーザー **AccountAadmin** を作成して、ユーザーのセキュリティ認証情報を書き留めます。

手順については、「IAM ユーザーガイド」の「AWS アカウント での IAM ユーザーの作成」を参照してください。

ii. フルアクセスを付与するユーザーポリシーをアタッチして、管理者に AccountAadmin を付与します。

手順については、「IAM ユーザーガイド」の「IAM ポリシーを管理する」を参照してください。

iii. AccountAadmin の IAM ユーザーサインイン URL を書き留めます。この URL は AWS Management Console にサインインする際に使用する必要があります。サインイン URL を見つける方法については、「IAM ユーザーガイド」の「Sign in to the AWS Management Console as an IAM user」を参照してください。URL はアカウントごとにメモしてください。

2. AWS CLI または AWS Tools for Windows PowerShell をセットアップします。管理者ユーザーの認証情報は以下のように保存します。

- AWS CLI を使用する場合は、設定ファイルに `AccountAadmin` プロファイルを作成します。
- AWS Tools for Windows PowerShell を使用する場合は、セッションの認証情報を `AccountAadmin` として保存します。

手順については、チュートリアル用のツールのセットアップ を参照してください。

ステップ 1: アカウント A でリソースを作成して、アクセス許可を付与する

アカウント A のユーザー `AccountAadmin` の認証情報と、IAM ユーザー専用のサインイン URL を使用して、AWS Management Consoleにサインインし、次の操作を行います。

1. バケットリソースと IAM ユーザーリソースの作成

a. Amazon S3 コンソールでバケットを作成します。バケットを作成した AWS リージョン を書き留めます。手順については、バケットの作成 を参照してください。

b. IAM コンソールで次の操作を行います。

i. Dave という名前のユーザーを作成します。

詳細な手順については、「IAM ユーザーガイド」の「IAM ユーザーの作成 (コンソール)」を参照してください。

ii. UserDave 認証情報を書き留めます。

iii. ユーザー Dave の Amazon リソースネーム (ARN) を書き留めます。IAM コンソールでユーザーを選択すると、[概要] タブにユーザー ARN が表示されます。

2. アクセス許可を付与します。

バケット所有者とユーザーが属する親アカウントが同じであるため、AWS アカウントは、バケットポリシー、ユーザーポリシー、またはその両方を使用してユーザーに許可を付与できます。この例では、両方を使用します。オブジェクトの所有者も同じアカウントである場合は、バケット所有者はバケットポリシー (または IAM ポリシー) でオブジェクトのアクセス権を付与することもできます。

a. Amazon S3 コンソールで、次のバケットポリシーを *awsexamplebucket1* にアタッチします。

このポリシーには 2 つのステートメントがあります。

- 最初のステートメントは、バケットオペレーションのアクセス許可 s3:GetBucketLocation と s3:ListBucket を Dave に付与します。
- 2 番目のステートメントは、アクセス許可 s3:GetObject を付与します。アカウント A はオブジェクトの所有者でもあるので、アカウント管理者はアクセス許可 s3:GetObject を付与できます。

Principal ステートメントでは、Dave はユーザー ARN によって識別されます。ポリシーエレメントの詳細については、Amazon S3 のポリシーとアクセス許可 を参照してください。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "statement1",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
```

```
            },
            "Action": [
               "s3:GetBucketLocation",
               "s3:ListBucket"
            ],
            "Resource": [
               "arn:aws:s3:::awsexamplebucket1"
            ]
         },
         {
            "Sid": "statement2",
            "Effect": "Allow",
            "Principal": {
               "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
               "arn:aws:s3:::awsexamplebucket1/*"
            ]
         }
   ]
}
```

b. 以下のポリシーを使用して、ユーザー Dave のインラインポリシーを作成します。このポリシーは、s3:PutObject アクセス許可を Dave に付与します。バケット名を指定してポリシーを更新する必要があります。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "PermissionForObjectOperations",
         "Effect": "Allow",
         "Action": [
            "s3:PutObject"
         ],
         "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*"
         ]
      }
   ]
```

```
}
```

手順については、「IAM ユーザーガイド」の「IAM ポリシーを管理する」を参照してください。コンソールにサインインするには、アカウント A の認証情報を使用する必要があります。

ステップ 2: アクセス許可をテストする

Dave の認証情報を使用して、アクセス許可が機能することを確認します。次の 2 つの手順のいずれかを使用できます。

AWS CLI を使用してアクセス許可をテストする

1. 次の UserDaveAccountA プロファイルを追加して、AWS CLI 設定ファイルを更新します。詳細については、「チュートリアル用のツールのセットアップ」を参照してください。

```
[profile UserDaveAccountA]
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. Dave がユーザーポリシーで許可されたオペレーションを実行できることを確認します。次の AWS CLI put-object コマンドを使用して、サンプルのオブジェクトをアップロードします。

   アップロードするソースファイルは、コマンドの --body パラメータで指定します。例えば、ファイルが Windows 端末の C: ドライブのルートにある場合、c:\HappyFace.jpg と指定します。--key パラメータは、オブジェクトのキー名を指定します。

```
aws s3api put-object --bucket awsexamplebucket1 --key HappyFace.jpg --
body HappyFace.jpg --profile UserDaveAccountA
```

   次の AWS CLI コマンドを実行して、オブジェクトを取得します。

```
aws s3api get-object --bucket awsexamplebucket1 --key HappyFace.jpg OutputFile.jpg
 --profile UserDaveAccountA
```

AWS Tools for Windows PowerShell を使用してアクセス許可をテストする

1. Dave の認証情報を AccountADave という名前で保存します。次に、この認証情報を使用して、オブジェクトの PUT と GET を実行します。

```
set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
 AccountADave
```

2. 保存したユーザー Dave の認証情報を使用して、AWS Tools for Windows PowerShell `Write-S3Object` コマンドでサンプルオブジェクトをアップロードします。

```
Write-S3Object -bucketname awsexamplebucket1 -key HappyFace.jpg -file HappyFace.jpg
 -StoredCredentials AccountADave
```

先にアップロードしたオブジェクトをダウンロードします。

```
Read-S3Object -bucketname awsexamplebucket1 -key HappyFace.jpg -file Output.jpg -
StoredCredentials AccountADave
```

例 2: バケット所有者がクロスアカウントのバケットのアクセス許可を付与する

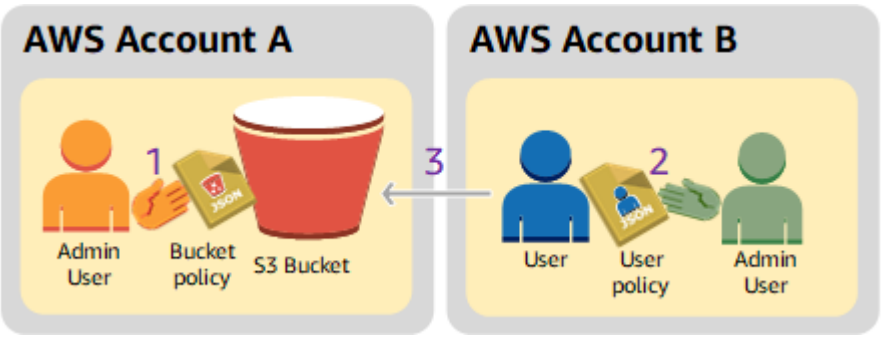
> **Important**
>
> アクセス許可は、個々のユーザーではなく、IAM ロールに付与することをお勧めします。これを行う方法については、クロスアカウントのアクセス許可の理解と IAM ロールの使用を参照してください。

トピック

ある AWS アカウント (例えば、アカウント A) で、バケットやオブジェクトなど、そのリソースにアクセスするためのアクセス許可を、別の AWS アカウント (アカウント B) に付与できます。アカウント B では、そのアカウントのユーザーに、付与されたアクセス許可を委任できます。この例のシナリオでは、バケット所有者が、特定のバケットオペレーションを実行するためのクロスアカウントアクセス許可を別のアカウントに付与します。

> **Note**
>
> また、バケットポリシーを使用して、アカウント A から直接アカウント B のユーザーにアクセス許可を付与することもできます。ただし、このユーザーには、アカウント B がアカウント A からアクセス許可を付与されていない場合でも、ユーザーが属している親アカウントであるアカウント B からのアクセス許可が必要です。このユーザーがリソース所有者と親アカウントの両方からのアクセス許可を持っている場合にのみ、ユーザーはリソースにアクセスできます。

この手順の概要を以下に示します。



1. アカウント A の管理者ユーザーは、特定のバケットオペレーションを実行するための、クロスアカウントアクセス許可を付与するバケットポリシーをアカウント B にアタッチします。

   アカウント B の管理者ユーザーは、自動的にこのアクセス許可を継承することに注意してください。
2. アカウント B の管理者ユーザーは、アカウント A から受け取ったアクセス許可を委任するユーザーポリシーをユーザーにアタッチします。
3. アカウント B のユーザーは、アカウント A が所有するバケットのオブジェクトにアクセスすることにより、アクセス許可を確認します。

この例では、2 個のアカウントが必要です。次の表に、これらのアカウントとそれぞれの管理者ユーザーの呼び方を示します。IAM ガイドライン (「管理者ユーザーを使用したリソースの作成とアクセス許可の付与について」を参照) に従い、このチュートリアルではルートユーザー認証情報は使用しません。その代わりに、各アカウントで管理者ユーザーを作成し、その認証情報を使用してリソースを作成し、アクセス許可を付与します。

AWS アカウント ID	アカウントの呼び方	アカウントの管理者ユーザー
1111-1111-1111	アカウント A	AccountAadmin
2222-2222-2222	アカウント B	AccountBadmin

ユーザーを作成し、アクセス許可を付与するためのすべてのタスクは、AWS Management Console で実行します。このチュートリアルでは、許可を検証するために、コマンドラインツールの AWS Command Line Interface (CLI) と AWS Tools for Windows PowerShell を使用するので、コードを記述する必要はありません。

チュートリアルの準備をする

1. 前のセクションの表に示したように、2 個の AWS アカウントがあり、各アカウントに 1 人ずつの管理者ユーザーが存在することを確認します。

   a. 必要な場合、AWS アカウントにサインアップします。

   b. アカウント A の認証情報を使用して、IAM コンソールにサインインし、管理者ユーザーを作成します。

      i. ユーザー **AccountAadmin** を作成して、セキュリティ認証情報を書き留めます。手順については、IAM ユーザーガイドの「AWS アカウントでの IAM ユーザーの作成」を参照してください。

      ii. フルアクセスを付与するユーザーポリシーをアタッチして、管理者に AccountAadmin を付与します。手順については、IAM ユーザーガイドの IAM ポリシーの管理を参照してください。

   c. IAM コンソールの [ダッシュボード] の IAM ユーザーサインイン URL をメモします このアカウントのすべてのユーザーは、AWS Management Console にサインインするときにこの URL を使用する必要があります。

      詳細については、IAM ユーザーガイドのユーザーがお客様のアカウントにサインインする方法 を参照してください。

   d. 前述のステップを、アカウント B の認証情報を使用してもう一度実行し、管理者ユーザー **AccountBadmin** を作成します。

2. AWS Command Line Interface (AWS CLI) または AWS Tools for Windows PowerShell をセットアップします。管理者ユーザーの認証情報は以下のように保存します。

   - AWS CLI を使用する場合は、設定ファイルに `AccountAadmin` と `AccountBadmin` の 2 つのプロファイルを作成します。
   - AWS Tools for Windows PowerShell を使用する場合は、セッションの認証情報を `AccountAadmin` および `AccountBadmin` として保存します。

   手順については、チュートリアル用のツールのセットアップ を参照してください。

3. プロファイルとも呼ばれる、管理者ユーザーの認証情報を保存します。入力する各コマンドで認証情報を指定する代わりにプロファイル名を使用できます。詳細については、「チュートリアル用のツールのセットアップ」を参照してください。

a. 2 つのアカウントの管理者ユーザー (AccountAadmin および AccountBadmin) のそれぞれの AWS CLI 認証情報ファイルにプロファイルを追加します。

```
[AccountAadmin]
aws_access_key_id = access-key-ID
aws_secret_access_key = secret-access-key
region = us-east-1

[AccountBadmin]
aws_access_key_id = access-key-ID
aws_secret_access_key = secret-access-key
region = us-east-1
```

b. AWS Tools for Windows PowerShell を使用している場合は、次のコマンドを実行します。

```
set-awscredentials -AccessKey AcctA-access-key-ID -SecretKey AcctA-secret-
access-key -storeas AccountAadmin
set-awscredentials -AccessKey AcctB-access-key-ID -SecretKey AcctB-secret-
access-key -storeas AccountBadmin
```

ステップ 1: アカウント A のタスクを実行する

ステップ 1.1: AWS Management Consoleにサインインする

アカウント A で IAM ユーザーのサインイン URL を使用して、まず AccountAadmin ユーザーとして AWS Management Console にサインインします。このユーザーはバケットを作成し、ポリシーをアタッチします。

ステップ 1.2: バケットを作成する

1. Amazon S3 コンソールでバケットを作成します。この演習では、米国東部 (バージニア北部) AWS リージョン に *DOC-EXAMPLE-BUCKET* という名前でバケットを作成することを前提とします。

   手順については、バケットの作成 を参照してください。

2. サンプルオブジェクトをバケットにアップロードします。

   方法については、ステップ 2: バケットにオブジェクトをアップロードする を参照してください。

ステップ 1.3: クロスアカウントアクセス許可を付与するバケットポリシーをアカウント B にアタッチする

このバケットポリシーは、s3:GetLifecycleConfiguration および s3:ListBucket アクセス許可をアカウント B に付与します。これは、AccountAadmin ユーザーの認証情報を使用してコンソールにサインインしていることを前提としています。

1. 次のバケットポリシーを *DOC-EXAMPLE-BUCKET* にアタッチします。このポリシーは、アカウント B に s3:GetLifecycleConfiguration および s3:ListBucket アクションのアクセス許可を付与します。

   手順については、Amazon S3 コンソールを使用したバケットポリシーの追加 を参照してください。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:GetLifecycleConfiguration",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
         ]
      }
   ]
}
```

2. アカウント B (とその管理ユーザー) がオペレーションを実行できることを確認します。

   - AWS CLI を使用して検証する

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET --profile AccountBadmin
aws s3api get-bucket-lifecycle-configuration --bucket DOC-EXAMPLE-BUCKET --
profile AccountBadmin
```

- AWS Tools for Windows PowerShell を使用して検証する

```
get-s3object -BucketName DOC-EXAMPLE-BUCKET -StoredCredentials AccountBadmin
get-s3bucketlifecycleconfiguration -BucketName DOC-EXAMPLE-BUCKET -
StoredCredentials AccountBadmin
```

ステップ 2: アカウント B のタスクを実行する

次に、アカウント B の管理者はユーザー Dave を作成し、アカウント A から受け取ったアクセス許可を委任します。

ステップ 2.1: AWS Management Consoleにサインインする

アカウント B の IAM ユーザーのサインイン URL を使用し、AccountBadmin ユーザーとして AWS Management Console にサインインします。

ステップ 2.2: アカウント B でユーザー Dave を作成する

IAM コンソールで、ユーザー **Dave** を作成します。

手順については、「IAM ユーザーガイド」の「IAM ユーザーの作成 (コンソール)」を参照してください。

ステップ 2.3: ユーザー Dave に許可を委任する

以下のポリシーを使用して、ユーザー Dave のインラインポリシーを作成します。バケット名を指定してポリシーを更新する必要があります。

これは、AccountBadmin ユーザーの認証情報を使用してコンソールにサインインしていることを前提としています。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example",
         "Effect": "Allow",
         "Action": [
            "s3:ListBucket"
         ],
```

```
         "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
         ]
      }
   ]
}
```

手順については、「IAM ユーザーガイド」の「IAM ポリシーを管理する」を参照してください。

ステップ 2.4: アクセス許可をテストする

これで、アカウント B の Dave はアカウント A が所有する *DOC-EXAMPLE-BUCKET* の内容をリストできます。次のいずれかの手順でアクセス許可を確認できます。

AWS CLI を使用してアクセス許可をテストする

1. AWS CLI 設定ファイルに UserDave プロファイルを追加します。設定ファイルの詳細については、チュートリアル用のツールのセットアップ を参照してください。

   ```
   [profile UserDave]
   aws_access_key_id = access-key
   aws_secret_access_key = secret-access-key
   region = us-east-1
   ```

2. コマンドプロンプトで、次の AWS CLI コマンドを入力して、Dave がアカウント A によって所有される *DOC-EXAMPLE-BUCKET* からオブジェクトリストを取得できることを検証します。このコマンドでは、UserDave プロファイルを指定していることに注意してください。

   ```
   aws s3 ls s3://DOC-EXAMPLE-BUCKET --profile UserDave
   ```

   Dave には他のアクセス許可はありません。したがって、他のオペレーション (次の get-bucket-lifecycle 設定など) を実行しようとすると、Amazon S3 はアクセス許可の拒否を返します。

   ```
   aws s3api get-bucket-lifecycle-configuration --bucket DOC-EXAMPLE-BUCKET --profile
    UserDave
   ```

AWS Tools for Windows PowerShell を使用してアクセス許可をテストする

1. Dave の認証情報を AccountBDave という名前で保存します。

```
set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
 AccountBDave
```

2. List Bucket コマンドを試してみます。

```
get-s3object -BucketName DOC-EXAMPLE-BUCKET -StoredCredentials AccountBDave
```

Dave には他のアクセス許可はありません。したがって、他のオペレーション (次の get-s3bucketlifecycleconfiguration など) を実行しようとすると、Amazon S3 はアクセス許可の拒否を返します。

```
get-s3bucketlifecycleconfiguration -BucketName DOC-EXAMPLE-BUCKET -
StoredCredentials AccountBDave
```

ステップ 3: (オプション) 明示的な拒否を試す

アクセスコントロールリスト (ACL)、バケットポリシー、またはユーザーポリシーを使用して、アクセス許可を付与できます。ただし、バケットポリシーまたはユーザーポリシーによって明示的な拒否が設定されている場合、他のアクセス許可よりも明示的な拒否が優先されます。テストのために、バケットポリシーを更新し、アカウント B の s3:ListBucket アクセス許可を明示的に拒否してみます。このポリシーでは、s3:ListBucket アクセス許可も付与されます。ただし、明示的な拒否が優先され、アカウント B またはアカウント B のユーザーは *DOC-EXAMPLE-BUCKET* 内のオブジェクトを一覧表示を行うことはできません。

1. アカウント A のユーザー AccountAadmin の認証情報を使用して、バケットポリシーを次のように置き換えます。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:GetLifecycleConfiguration",
```

```
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
         ]
      },
      {
         "Sid": "Deny permission",
         "Effect": "Deny",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
         ]
      }
   ]
}
```

2. ここで AccountBadmin の認証情報を使用してバケットリストを取得しようとすると、アクセスは拒否されます。

   - AWS CLI を使用して、次のコマンドを実行します。

     ```
     aws s3 ls s3://DOC-EXAMPLE-BUCKET --profile AccountBadmin
     ```

   - AWS Tools for Windows PowerShell を使用して、次のコマンドを実行します。

     ```
     get-s3object -BucketName DOC-EXAMPLE-BUCKET -StoredCredentials AccountBDave
     ```

ステップ 4: クリーンアップする

1. テストが終了したら、次の手順でクリーンアップを行います。

   - アカウント A の認証情報を使用して AWS Management Console (AWS Management Console) にサインインし、次の操作を行います。

- Amazon S3 コンソールで、*DOC-EXAMPLE-BUCKET* にアタッチされているバケットポリシーを削除します。バケットの [プロパティ] で、[アクセス許可] セクションのポリシーを削除します。
- バケットをこの演習のために作成した場合は、Amazon S3 コンソールでオブジェクトを削除してから、バケットを削除します。
- IAM コンソールで、AccountAadmin ユーザーを削除します。

2. アカウント B の認証情報を使用して IAM コンソールにサインインします。AccountBadmin ユーザーを削除します。手順については、「IAM ユーザーガイド」の「IAM ユーザーの削除」を参照してください。

例 3: バケット所有者が自分の所有していないオブジェクトに対するアクセス許可を付与する

> **Important**
>
> アクセス許可は、個々のユーザーではなく、IAM ロールに付与することをお勧めします。これを行う方法については、クロスアカウントのアクセス許可の理解と IAM ロールの使用を参照してください。

トピック

- ステップ 0: チュートリアルの準備をする
- ステップ 1: アカウント A のタスクを実行する
- ステップ 2: アカウント B のタスクを実行する
- ステップ 3: アクセス許可をテストする
- ステップ 4: クリーンアップする

この例のシナリオでは、バケット所有者はオブジェクトへのアクセス許可を付与しますが、バケット内のオブジェクトの一部はバケット所有者によって所有されていません。この例では、バケット所有者は、自分のアカウントのユーザーにアクセス許可を付与しようとしています。

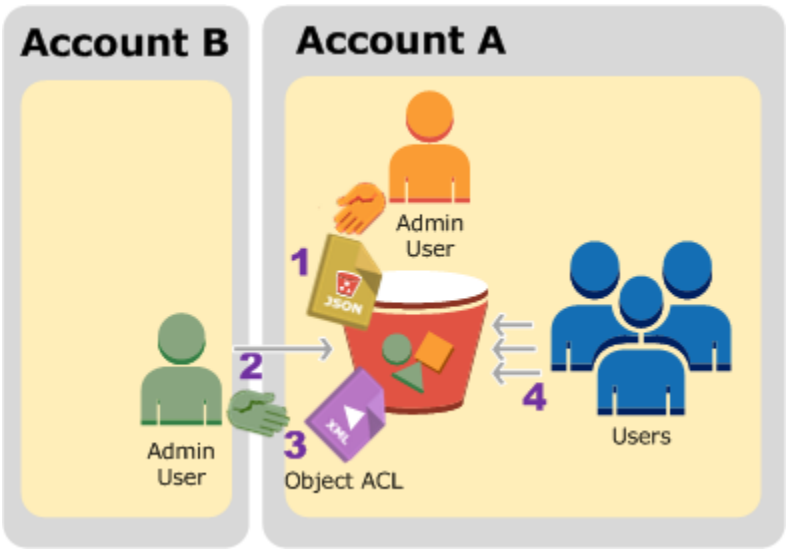
バケット所有者は、他の AWS アカウントにオブジェクトのアップロードを許可できます。デフォルトでは、バケット所有者は別の AWS アカウント によってバケットに書き込まれたオブジェクトを所有しません。オブジェクトは、S3 バケットに書き込むアカウントによって所有されています。バケット所有者がバケット内のオブジェクトを所有していない場合、オブジェクト所有者は最初にオブ

ジェクトのアクセスコントロールリスト (ACL) を使用してバケット所有者に権限を付与する必要があります。その後、バケット所有者は所有していないオブジェクトのアクセス許可を付与できます。詳細については、「Amazon S3 のバケットとオブジェクトの所有権」を参照してください。

バケット所有者がバケットの S3 オブジェクト所有権にバケット所有者強制設定を適用すると、バケット所有者は、別の AWS アカウント によって書き込まれたオブジェクトを含む、バケット内のすべてのオブジェクトを所有します。これにより、オブジェクトがバケット所有者によって所有されていないという問題が解決されます。次に、自分のアカウントまたは他の AWS アカウント アカウントのユーザーにアクセス許可を委任できます。

> **Note**
>
> S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、バケットにアップロードされる新しいオブジェクト所有権を制御し、ACL を無効にするのに使用できます。デフォルトでは、オブジェクト所有権はバケット所有者の強制設定に設定され、すべての ACL は無効になります。ACL を無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。
> Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。ACL を無効にすると、誰がオブジェクトをバケットにアップロードしたかに関係なく、ポリシーを使用してバケット内のすべてのオブジェクトへのアクセスを制御できます。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

この例では、バケット所有者がオブジェクト所有権のバケット所有者強制設定を適用していないと仮定します。バケット所有者は、自分のアカウントのユーザーに許可を委任します。この手順の概要を以下に示します。



1. アカウント A の管理者ユーザーが、2 つのステートメントを持つバケットポリシーをアタッチします。
   - オブジェクトをアップロードするクロスアカウントアクセスをアカウント B に許可します。
   - 自分のアカウントのユーザーにバケット内のオブジェクトへのアクセスを許可します。
2. アカウント B の管理者ユーザーは、アカウント A が所有するバケットにオブジェクトをアップロードします。
3. アカウント B の管理者は、オブジェクト ACL を更新して、オブジェクトに対するフルコントロールアクセス許可をバケット所有者に付与します。
4. アカウント A のユーザーは、所有者に関わりなくバケット内のオブジェクトにアクセスできることを確認します。

この例では、2 個のアカウントが必要です。次の表に、これらのアカウントの呼び方とそれぞれの管理者ユーザーを示します。このチュートリアルでは、推奨される IAM ガイドラインに従って、アカウントのルートユーザー認証情報を使用しません。詳細については、「管理者ユーザーを使用したリソースの作成とアクセス許可の付与について」を参照してください。その代わりに、各アカウントに管理者を作成し、その認証情報を使用してリソースを作成し、アクセス許可を付与します。

AWS アカウント ID	アカウントの呼び方	アカウントの管理者
1111-1111-1111	アカウント A	AccountAadmin
2222-2222-2222	アカウント B	AccountBadmin

ユーザーを作成し、アクセス許可を付与するためのすべてのタスクは、AWS Management Console で実行します。このチュートリアルでは、許可を検証するために、コマンドラインツールの AWS Command Line Interface (AWS CLI) と AWS Tools for Windows PowerShell を使用するので、コードを記述する必要はありません。

ステップ 0: チュートリアルの準備をする

1. 前のセクションの表に示したように、2 個の AWS アカウントがあり、各アカウントに 1 人ずつの管理者が存在することを確認します。

   a. 必要な場合、AWS アカウントにサインアップします。

   b. アカウント A の認証情報を使用して、IAM コンソールにサインインし、次の操作を行って管理者ユーザーを作成します。

      - ユーザー **AccountAadmin** を作成し、ユーザーのセキュリティ認証情報を書き留めます。ユーザーの追加の詳細については、「IAM ユーザーガイド」の「AWS アカウント での IAM ユーザーの作成」を参照してください。
      - AccountAadmin にフルアクセスを許可するユーザーポリシーをアタッチして、管理者権限を付与します。手順については、「IAM ユーザーガイド」の「IAM ポリシーを管理する」を参照してください。
      - IAM コンソールの [ダッシュボード] で、[IAM ユーザーのサインイン URL] を書き留めます。このアカウントのユーザーは、この URL を使用して AWS Management Console にサインインします。詳細については、IAM ユーザーガイドのユーザーがお客様のアカウントにサインインする方法 を参照してください。

   c. 前述のステップを、アカウント B の認証情報を使用してもう一度実行し、管理者ユーザー **AccountBadmin** を作成します。

2. AWS CLI または Tools for Windows PowerShell をセットアップします。管理者認証情報を必ず次のように保存してください。

   - AWS CLI を使用する場合は、設定ファイルに `AccountAadmin` と `AccountBadmin` の 2 つのプロファイルを作成します。
   - Tools for Windows PowerShell を使用する場合、セッションの認証情報を `AccountAadmin` および `AccountBadmin` として保存します。

   手順については、チュートリアル用のツールのセットアップ を参照してください。

ステップ 1: アカウント A のタスクを実行する

アカウント A に対して、次の手順を実行します。

ステップ 1.1: コンソールにサインインする

アカウント A の IAM ユーザーのサインイン URL を使用して、**AccountAadmin** ユーザーとして AWS Management Console にサインインします。このユーザーはバケットを作成し、ポリシーをアタッチします。

ステップ 1.2: バケットとユーザーを作成し、ユーザーにアクセス許可を付与するバケットポリシーを追加する

1. Amazon S3 コンソールでバケットを作成します。この演習では、米国東部 (バージニア北部) AWS リージョン に *example-s3-bucket1* という名前でバケットを作成することを前提とします。

   手順については、バケットの作成 を参照してください。
2. IAM コンソールで、ユーザー **Dave** を作成します。

   詳細な手順については、「IAM ユーザーガイド」の「IAM ユーザーの作成 (コンソール)」を参照してください。
3. ユーザー Dave の認証情報を書き留めます。
4. Amazon S3 コンソールで、次のバケットポリシーを *example-s3-bucket1* バケットにアタッチします。手順については、Amazon S3 コンソールを使用したバケットポリシーの追加 を参照してください。バケットポリシーを追加するには、このステップに従います。アカウント ID を確認する方法については、AWS アカウント ID の検索 を参照してください。

   このポリシーは、アカウント B に s3:PutObject および s3:ListBucket アクセス許可を付与します。このポリシーはまた、ユーザー Dave に s3:GetObject アクセス許可を付与します。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Statement1",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
```

```
            "Action": [
               "s3:PutObject",
               "s3:ListBucket"
            ],
            "Resource": [
               "arn:aws:s3:::example-s3-bucket1/*",
               "arn:aws:s3:::example-s3-bucket1"
            ]
         },
         {
            "Sid": "Statement3",
            "Effect": "Allow",
            "Principal": {
               "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": [
               "s3:GetObject"
            ],
            "Resource": [
               "arn:aws:s3:::example-s3-bucket1/*"
            ]
         }
      ]
}
```

ステップ 2: アカウント B のタスクを実行する

これでアカウント B はアカウント A のバケットに対するオペレーション実行のアクセス許可を得ましたので、管理者は次の作業を行います。

- アカウント A のバケットにオブジェクトをアップロードします。
- バケット所有者であるアカウント A にそのバケットへのフルコントロールをオブジェクトの ACL で付与します。

AWS CLI の使用

1. put-object AWS CLI コマンドを使用して、オブジェクトをアップロードします。アップロードするソースファイルは、コマンドの --body パラメータで指定します。例えば、ファイルが Windows 端末の C: ドライブにある場合、c:\HappyFace.jpg のように指定します。--key パラメータは、オブジェクトのキー名を指定します。

```
aws s3api put-object --bucket example-s3-bucket1 --key HappyFace.jpg --body
 HappyFace.jpg --profile AccountBadmin
```

2. バケット所有者に対するオブジェクトのフルコントロールの許可をオブジェクト ACL に追加します。正規ユーザー ID を検索する方法については、「AWS Account Management リファレンスガイド」の「AWS アカウント の正規ユーザー ID を検索する」を参照してください。

```
aws s3api put-object-acl --bucket example-s3-bucket1 --key HappyFace.jpg --grant-
full-control id="AccountA-CanonicalUserID" --profile AccountBadmin
```

Tools for Windows PowerShell の使用

1. `Write-S3Object` コマンドを使用して、オブジェクトをアップロードします。

```
Write-S3Object -BucketName example-s3-bucket1 -key HappyFace.jpg -file
 HappyFace.jpg -StoredCredentials AccountBadmin
```

2. バケット所有者に対するオブジェクトのフルコントロールの許可をオブジェクト ACL に追加します。

```
Set-S3ACL -BucketName example-s3-bucket1 -Key HappyFace.jpg -CannedACLName "bucket-
owner-full-control" -StoredCreden
```

ステップ 3: アクセス許可をテストする

アカウント A のユーザー Dave が、アカウント B が所有するオブジェクトにアクセスできることを確認します。

AWS CLI の使用

1. ユーザー Dave の認証情報を AWS CLI の設定ファイルに追加して、新しいプロファイル `UserDaveAccountA` を作成します。詳細については、「チュートリアル用のツールのセットアップ」を参照してください。

```
[profile UserDaveAccountA]
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
```

```
region = us-east-1
```

2. get-object CLI コマンドを実行して HappyFace.jpg をダウンロードし、ローカルに保存します。--profile パラメータを追加して、ユーザー Dave の認証情報を指定します。

```
aws s3api get-object --bucket example-s3-bucket1 --key HappyFace.jpg Outputfile.jpg
 --profile UserDaveAccountA
```

Tools for Windows PowerShell の使用

1. ユーザー Dave の AWS 認証情報を、UserDaveAccountA という名前で永続的ストアに保存します。

```
Set-AWSCredentials -AccessKey UserDave-AccessKey -SecretKey UserDave-
SecretAccessKey -storeas UserDaveAccountA
```

2. Read-S3Object コマンドを実行して HappyFace.jpg オブジェクトをダウンロードし、ローカルに保存します。-StoredCredentials パラメータを追加して、ユーザー Dave の認証情報を指定します。

```
Read-S3Object -BucketName example-s3-bucket1 -Key HappyFace.jpg -file HappyFace.jpg
  -StoredCredentials UserDaveAccountA
```

ステップ 4: クリーンアップする

1. テストが終了したら、次の手順でクリーンアップを行います。

   - アカウント A の認証情報を使用して AWS Management Console にサインインし、次の操作を行います。

     - Amazon S3 コンソールで、*example-s3-bucket1* にアタッチされているバケットポリシーを削除します。バケットの [プロパティ] で、[アクセス許可] セクションのポリシーを削除します。
     - バケットをこの演習のために作成した場合は、Amazon S3 コンソールでオブジェクトを削除してから、バケットを削除します。
     - IAM コンソールで、[AccountAadmin] ユーザーを削除します。手順については、「IAM ユーザーガイド」の「IAM ユーザーの削除」を参照してください。

2. アカウント B の認証情報を使用して AWS Management Console にサインインします。IAM コンソールで、ユーザー [AccountBadmin] を削除します。

例 4 - バケット所有者が所有権のないオブジェクトへのクロスアカウントアクセス許可を付与する

トピック

この例のシナリオでは、バケット所有者が、オブジェクトをアップロードするためのアクセス許可を他の AWS アカウント に付与します。バケットの S3 オブジェクト所有権にバケット所有者強制設定を適用した場合、別の AWS アカウント によって書き込まれたオブジェクトを含む、バケット内のすべてのオブジェクトを所有します。これにより、バケット所有者であるユーザーがオブジェクトを所有していないという問題が解決されます。次に、自分のアカウントまたは他の AWS アカウントアカウントのユーザーにアクセス許可を委任できます。S3 オブジェクト所有権のバケット所有者強制設定が有効になっていないとします。つまり、バケットは、他の AWS アカウントが所有するオブジェクトを含むことができます。

バケット所有者は、オブジェクトの所有者が誰であるかに関係なく、オブジェクトに対するクロスアカウントのアクセス許可を別のアカウントのユーザーに付与しなければならない場合があります。例えば、課金アプリケーションがユーザーとしてオブジェクトのメタデータにアクセスしなければならない場合などです。ここで、主に 2 つの問題が生じます。

- バケット所有者には、他の AWS アカウントで作成されたオブジェクトに対する許可はありません。バケット所有者が、自分が所有していないオブジェクトに対するアクセス許可を付与するには、オブジェクトの所有者 (オブジェクトを作成した) は、最初にバケット所有者にアクセス許可を付与する必要があります。この所有者は、オブジェクトを作成した AWS アカウントです。これで、バケット所有者は、それらのアクセス許可を委任できるようになります。
- バケット所有者は、自分のアカウントのユーザーに許可を委任できます (「例 3: バケット所有者が自分の所有していないオブジェクトに対するアクセス許可を付与する」を参照)。ただし、クロス

アカウントの委任はサポートされていないため、バケット所有者アカウントは他の AWS アカウント にアクセス許可を委任できません。

このシナリオでは、バケット所有者はオブジェクトへのアクセス許可を持つ AWS Identity and Access Management (IAM) ロールを作成できます。その後、バケット所有者は別の AWS アカウント にロールを引き受けるアクセス許可を付与して、バケットのオブジェクトへのアクセスを一時的に有効化できます。

> **Note**
>
> S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、バケットにアップロードされる新しいオブジェクト所有権を制御し、ACL を無効にするのに使用できます。デフォルトでは、オブジェクト所有権はバケット所有者の強制設定に設定され、すべての ACL は無効になります。ACL を無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。
> Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。ACL を無効にすると、誰がオブジェクトをバケットにアップロードしたかに関係なく、ポリシーを使用してバケット内のすべてのオブジェクトへのアクセスを制御できます。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

クロスアカウントのアクセス許可の理解と IAM ロールの使用

IAM ロールは、リソースへのアクセスを委任する、いくつかのシナリオで使用されます。なかでも、クロスアカウントアクセスは重要なシナリオです。この例では、バケット所有者 (アカウント A) は、IAM ロールを使用して、他の AWS アカウント (アカウント C) のユーザーに一時的にオブジェクトへのクロスアカウントの許可を委任します。作成された IAM ロールには、次の 2 つのポリシーがアタッチされています。

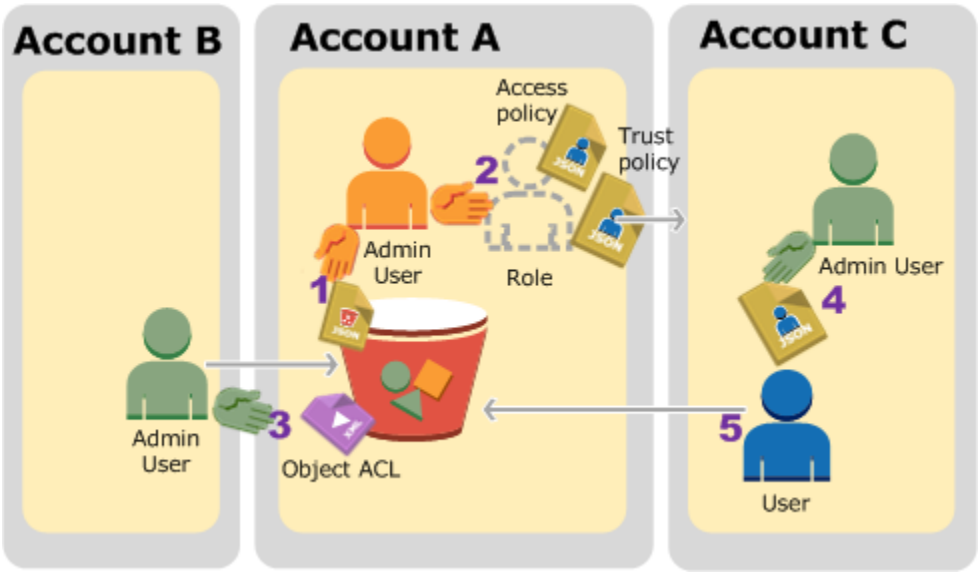
- ロールを引き受ける別の AWS アカウントを指定する信頼ポリシー。
- ロールを引き受けるユーザーに付与されるアクセス許可 (`s3:GetObject` など) を定義するアクセスポリシー。ポリシーで指定できるアクセス許可のリストについては、Amazon S3 のポリシーアクション を参照してください。

信頼ポリシーで指定された AWS アカウントは、アカウントに属するユーザーにロールを引き受ける許可を付与します。ユーザーは、以下の手順でオブジェクトにアクセスします。

- ロールを引き受け、その応答として一時的なセキュリティ認証情報を取得します。
- 一時的なセキュリティ認証情報を使用して、バケットのオブジェクトにアクセスします。

IAM ロールの詳細については、「IAM ユーザーガイド」の「IAM ロール」を参照してください。

この手順の概要を以下に示します。



1. アカウント A の管理者ユーザーは、バケットポリシーをアタッチし、オブジェクトをアップロードする条件付きのアクセス許可をアカウント B に付与します。
2. アカウント A の管理者ユーザーは、IAM ロールを作成し、アカウント C との間に信頼を確立します。これで、アカウント C のユーザーはアカウント A にアクセスできるようになります。ロールにアタッチされたアクセスポリシーは、アカウント C のユーザーがアカウント A にアクセスするときに実行できる操作を制限します。
3. アカウント B の管理者は、アカウント A が所有するバケットにオブジェクトをアップロードし、バケット所有者にフルコントロールのアクセス許可を付与します。
4. アカウント C の管理者は、ユーザーを作成し、ロールを引き受けるためのユーザーポリシーをアタッチします。
5. アカウント C のユーザーが初めてロールを引き受けると、一時的なセキュリティ認証情報が返されます。ユーザーは、この一時的な認証情報を使用して、バケットのオブジェクトにアクセスします。

この例では、3 つのアカウントが必要です。次の表に、これらのアカウントの呼び方とそれぞれの管理者ユーザーを示します。IAM のガイドライン (「管理者ユーザーを使用したリソースの作成とアクセス許可の付与について」を参照) に従い、このチュートリアルでは AWS アカウントのルートユーザーの認証情報を使用しません。その代わりに、各アカウントで管理者ユーザーを作成し、その認証情報を使用してリソースを作成し、アクセス許可を付与します。

AWS アカウント ID	アカウントの呼び方	アカウントの管理者ユーザー
1111-1111-1111	アカウント A	AccountAadmin
2222-2222-2222	アカウント B	AccountBadmin
3333-3333-3333	アカウント C	AccountCadmin

ステップ 0: チュートリアルの準備をする

> **Note**
>
> テキストエディタを開いて、手順を実行しながら、いくつかの情報を書き留めることができます。特に、コンソールに接続する各アカウントのアカウント ID、正規ユーザー ID、IAM ユーザーのサインイン URL、IAM ユーザーとロールの Amazon リソースネーム (ARN) が必要になります。

1. 前のセクションの表に示したように、3 つの AWS アカウント があり、各アカウントに 1 人ずつ管理者ユーザーが存在することを確認します。

   a. 必要に応じて AWS アカウントにサインアップします。ここでは、それぞれのアカウントをアカウント A、アカウント B、アカウント C と呼ぶことにします。

   b. アカウント A の認証情報を使用して、IAM コンソールにサインインし、次の操作を行って管理者ユーザーを作成します。

      - ユーザー **AccountAadmin** を作成して、そのセキュリティ認証情報を書き留めます。ユーザーの追加の詳細については、「IAM ユーザーガイド」の「AWS アカウント での IAM ユーザーの作成」を参照してください。

- フルアクセスを付与するユーザーポリシーをアタッチして、管理者に AccountAadmin を付与します。手順については、「IAM ユーザーガイド」の「IAM ポリシーを管理する」を参照してください。
- IAM コンソールの [ダッシュボード] で、[IAM ユーザーのサインイン URL] を書き留めます。このアカウントのユーザーは、この URL を使用して AWS Management Console にサインインします。詳細については、「IAM ユーザーガイド」の「Sign in to the AWS Management Console as an IAM user」を参照してください。

c. 前述の手順を繰り返し、アカウント B およびアカウント C の管理者ユーザーを作成します。

2. アカウント C で、正規ユーザー ID を書き留めます。

   アカウント A で IAM ロールを作成したら、信頼ポリシーでアカウント C のアカウント ID を指定して、ロールを引き受けるアクセス許可をアカウント C に付与します。アカウント情報は、次の手順で確認できます。

   a. AWS アカウント ID またはアカウントエイリアス、IAM ユーザー名、パスワードを使用して、Amazon S3 コンソールにサインインします。

   b. バケットの詳細を確認する Amazon S3 バケットの名前を選択します。

   c. [Permissions (アクセス許可)] タブを選択してから、[アクセスコントロールリスト] を選択します。

   d. AWS アカウント のアクセスのセッションで、アカウント 列は `c1daexampleaaf850ea79cf0430f33d72579fd1611c97f7ded193374c0b163b6` のような長い識別子です。これが正規ユーザー ID です。

3. バケットポリシーを作成するときは、次の情報が必要になります。これらの値を書き留めます。

   - アカウント A の正規ユーザー ID – アカウント A の管理者がアカウント B の管理者にオブジェクトをアップロードする条件付きのアクセス許可を付与するとき、オブジェクトに対するフルコントロールを取得するアカウント A のユーザーの正規ユーザー ID を指定します。

     > **Note**
     >
     > 正規ユーザー ID は、Amazon S3 固有の概念です。64 文字から成る、難読化されたアカウント ID です。

   - アカウント B 管理者の ユーザー ARN - ユーザー ARN は IAM コンソールにあります。[概要] タブでユーザーを選択し、ユーザーの ARN を見つけます。

バケットポリシーでは、オブジェクトをアップロードするアクセス許可を AccountBadmin に付与し、ARN を使用してユーザーを指定します。ARN 値の例を示します。

```
arn:aws:iam::AccountB-ID:user/AccountBadmin
```

4. AWS Command Line Interface (CLI) または AWS Tools for Windows PowerShell をセットアップします。管理者ユーザーの認証情報は以下のように保存します。

   - AWS CLI を使用する場合は、設定ファイルに AccountAadmin と AccountBadmin のプロファイルを作成します。
   - AWS Tools for Windows PowerShell を使用する場合は、セッションの認証情報を AccountAadmin および AccountBadmin として保存します。

   手順については、チュートリアル用のツールのセットアップ を参照してください。

ステップ 1: アカウント A のタスクを実行する

この例では、アカウント A はバケット所有者です。したがって、アカウント A のユーザー AccountAadmin は以下を実行します。

- バケットを作成します。
- オブジェクトをアップロードするアクセス許可をアカウント B の管理者に付与するバケットポリシーをアタッチします。
- バケット内のオブジェクトにアクセスできるように、ロールを引き受けるアクセス許可をアカウント C に付与する IAM ロールを作成します。

ステップ 1.1: AWS Management Consoleにサインインする

アカウント A で IAM ユーザーのサインイン URL を使用して、まず **AccountAadmin** ユーザーとして AWS Management Console にサインインします。このユーザーはバケットを作成し、ポリシーをアタッチします。

ステップ 1.2: バケットを作成してバケットポリシーをアタッチする

Amazon S3 コンソールで、次の操作を行います。

1. バケットを作成します。この演習では、バケットの名前を *example-s3-bucket1* とします。

手順については、バケットの作成 を参照してください。

2. 次のバケットポリシーをアタッチします。このポリシーは、オブジェクトをアップロードする条件付きのアクセス許可をアカウント B の管理者に付与します。

   example-s3-bucket1、*AccountB-ID*、*CanonicalUserId-of-AWSaccountA-BucketOwner* に独自の値を指定してポリシーを更新します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "111",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::example-s3-bucket1/*"
        },
        {
            "Sid": "112",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::example-s3-bucket1/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=CanonicalUserId-of-
AWSaccountA-BucketOwner"
                }
            }
        }
    ]
}
```

ステップ 1.3: アカウント C にクロスアカウントアクセスを許可する IAM ロールをアカウント A に作成する

IAM コンソールで、アカウント C にロールを引き受けるアクセス許可を付与する IAM ロール (**examplerole**) を作成します。このロールはアカウント A に作成する必要があるため、アカウント A の管理者としてサインインしていることを再度確認してください。

1. ロールを作成する前に、ロールに必要なアクセス許可を定義する管理ポリシーを準備します。後の手順で、このポリシーをロールにアタッチします。

   a. 左側のナビゲーションペインで [ポリシー] を選択し、続いて [ポリシーの作成] を選択します。

   b. [独自のポリシーを作成] の横で、[選択] を選択します。

   c. [Policy Name (ポリシー名)] フィールドに「**access-accountA-bucket** 」と入力します。

   d. 以下のアクセスポリシーをコピーして、[Policy Document (ポリシードキュメント)] フィールドに貼り付けます。このアクセスポリシーは、ロールに `s3:GetObject` のアクセス許可を付与するため、アカウント C のユーザーがこのロールを引き受けると、`s3:GetObject` オペレーションのみを実行できます。

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::example-s3-bucket1/*"
       }
     ]
   }
   ```

   e. [ポリシーの作成] を選択します。

   新しいポリシーが管理ポリシーの一覧に表示されます。

2. 左側のナビゲーションペインで、[ロール]、[新しいロールの作成] の順に選択します。

3. [ロールタイプの選択] で [クロスアカウントアクセスのロール] を選択して、[所有する AWS アカウント の間のアクセスの提供] の隣の [選択] ボタンを選択します。

4. アカウント C のアカウント ID を入力します。

このチュートリアルでは、ロールを引き受けるユーザーに多要素認証 (MFA) を必須とする必要がないため、このオプションは選択していない状態のままにしておきます。

5. [次のステップ] を選択して、そのロールに関連するアクセス許可を設定します。
6. 作成した access-accountA-bucket ポリシーの横にあるチェックボックスを選択し、[次のステップ] を選択します。

   [Review] ページが表示されます。このページで作成前のロールの設定を確認できます。このページで注意する重要な項目は、このロールを使用する必要があるユーザーに送信できるリンクです。ユーザーがこのリンクをクリックすると、[ロールの切り替え] ページがすぐに表示されます。このページには、[アカウント ID] と [ロール名] がすでに設定されています。このリンクは、クロスアカウントロールの [ロールの概要] ページで後で確認することもできます。
7. ロール名に「examplerole」と入力し、[次のステップ] を選択します。
8. ロールを確認したら、[ロールの作成] を選択します。

   examplerole ロールがロールの一覧に表示されます。
9. ロール名 examplerole を選択します。
10. [Trust Relationships (信頼関係)] タブを選択します。
11. [ポリシードキュメントの表示] を選択して、表示されている信頼ポリシーが次のポリシーと一致するか確認します。

    次の信頼ポリシーは、アカウント C との信頼を確立し、sts:AssumeRole アクションを許可します。詳細については、AWS Security Token Service API リファレンスの「AssumeRole」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountC-ID:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

12. 作成した examplerole ロールの Amazon リソースネーム (ARN) を書き留めます。

    後の手順で、このロールの引き受けを許可するユーザーポリシーを IAM ユーザーにアタッチしますが、ロールは ARN 値で指定します。

ステップ 2: アカウント B のタスクを実行する

アカウント A が所有するバケットの例では、他のアカウントが所有するオブジェクトが必要です。このステップでは、アカウント B の管理者は、コマンドラインツールを使用してオブジェクトをアップロードします。

- put-object AWS CLI コマンドを使用して、オブジェクトを *example-s3-bucket1* にアップロードします。

  ```
  aws s3api put-object --bucket example-s3-bucket1 --key HappyFace.jpg --
  body HappyFace.jpg --grant-full-control id="canonicalUserId-ofTheBucketOwner" --
  profile AccountBadmin
  ```

  次の点に注意してください。

  - --Profile パラメータは AccountBadmin のプロファイルを指定しているため、オブジェクトはアカウント B によって所有されます。
  - grant-full-control パラメータは、バケットポリシーに従って、バケット所有者にオブジェクトに対するフルコントロールのアクセス許可を付与します。
  - --body パラメータは、アップロードするソースファイルを指定します。例えば、ファイルが Windows コンピュータの C: ドライブにある場合は、「c:\HappyFace.jpg」と指定します。

ステップ 3: アカウント C のタスクを実行する

これまでの手順で、アカウント A はすでに examplerole ロールを作成し、アカウント C との信頼を確立しています。これにより、アカウント C のユーザーは、アカウント A にアクセスすることができます。このステップでは、アカウント C の管理者がユーザー (Dave) を作成し、アカウント A から取得した sts:AssumeRole のアクセス許可を委任します。examplerole を引き受けた Dave は、一時的にアカウント A へのアクセス権を取得します。アカウント A がロールにアタッチしたア

クセスポリシーは、アカウント A にアクセスする Dave が実行できるアクション (特に *example-s3-bucket1* のオブジェクトを取得すること) を制限します。

ステップ 3.1: アカウント C でユーザーを作成し、examplerole を引き受けるアクセス許可を委任する

1. アカウント C で IAM ユーザーのサインイン URL を使用して、まず **AccountCadmin** ユーザーとして AWS Management Console にサインインします。

2. IAM コンソールで、ユーザー Dave を作成します。

   詳細な手順については、「IAM ユーザーガイド」の「AWS Management Console での IAM ユーザーの作成」を参照してください。

3. Dave の認証情報を書き留めます。Dave は、`examplerole` ロールを引き受けるのに、これらの認証情報を必要とします。

4. アカウント A の `sts:AssumeRole` ロールで `examplerole` アクセス許可を Dave に委任する Dave IAM ユーザー用のインラインポリシーを作成します。

   a. 左側のナビゲーションペインで、[ユーザー] を選択します。

   b. ユーザー名 [Dave] を選択します。

   c. ユーザー詳細ページで [Permissions (アクセス許可)] タブを選択し、[Inline Policies (インラインポリシー)] セクションを展開します。

   d. [click here (ここをクリック)] (または [Create User Policy (ユーザーポリシーの作成)]) を選択します。

   e. [Custom Policy] を選択し、[Select] を選択します。

   f. [Policy Name (ポリシー名)] フィールドにポリシーの名前を入力します。

   g. 以下のポリシーをコピーして、[Policy Document (ポリシードキュメント)] フィールドに貼り付けます。

      AccountA-ID を指定してポリシーを更新する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole"],
      "Resource": "arn:aws:iam::AccountA-ID:role/examplerole"
```

```
    }
  ]
}
```

h. [ポリシーを適用] を選びます。

5. 別のプロファイル AccountCDave を追加して、Dave の認証情報を AWS CLI の設定ファイルに保存します。

```
[profile AccountCDave]
aws_access_key_id = UserDaveAccessKeyID
aws_secret_access_key = UserDaveSecretAccessKey
region = us-west-2
```

ステップ 3.2: ロール (examplerole) を引き受けてオブジェクトにアクセスする

Dave は、以下の手順でアカウント A が所有するバケットのオブジェクトにアクセスできます。

- Dave は、まず自分の認証情報を使用して、examplerole を引き受けます。一時的な認証情報が返されます。
- この一時的な認証情報を使用して、アカウント A のバケットのオブジェクトにアクセスします。

1. コマンドプロンプトで、AccountCDave プロファイルを使用して、次の AWS CLI assume-role コマンドを実行します。

   examplerole が定義されている *AccountA-ID* を指定して、コマンドの ARN 値を更新する必要があります。

```
aws sts assume-role --role-arn arn:aws:iam::AccountA-ID:role/examplerole --profile
 AccountCDave --role-session-name test
```

   AWS Security Token Service (AWS STS) が一時的なセキュリティ認証情報 (アクセスキー ID、シークレットアクセスキー、およびセッショントークン) を返します。

2. 一時的なセキュリティ認証情報を TempCred プロファイルの AWS CLI 設定ファイルに保存します。

```
[profile TempCred]
aws_access_key_id = temp-access-key-ID
aws_secret_access_key = temp-secret-access-key
```

```
aws_session_token = session-token
region = us-west-2
```

3. コマンドプロンプトで、一時的な認証情報を使用して、次の AWS CLI コマンドを実行してオブジェクトにアクセスします。例えば、コマンドで head-object API を指定すると、HappyFace.jpg オブジェクトのオブジェクトメタデータが取得されます。

```
aws s3api get-object --bucket example-s3-bucket1 --key HappyFace.jpg SaveFileAs.jpg
 --profile TempCred
```

examplerole にアタッチされているアクセスポリシーでは、アクションが許可されているため、Amazon S3 はリクエストを処理します。バケットの任意のオブジェクトに対して任意のアクションを試してみましょう。

例えば、get-object-acl のアクションを試すと、このロールではそのアクションは許可されていないため、アクセス許可が拒否されます。

```
aws s3api get-object-acl --bucket example-s3-bucket1 --key HappyFace.jpg --profile
 TempCred
```

この例では、ユーザー Dave がロールを引き受け、一時的な認証情報を使用してオブジェクトにアクセスしました。同様に、アカウント C のアプリケーションが *example-s3-bucket1* のオブジェクトにアクセスすることも可能です。アプリケーションが一時的なセキュリティ認証情報を取得すれば、アカウント C はこのアプリケーションに examplerole を引き受けるアクセス許可を委任できます。

ステップ 4: クリーンアップする

1. テストが終了したら、次の手順でクリーンアップを行います。

   - アカウント A の認証情報を使用して AWS Management Console にサインインし、次の操作を行います。

     - Amazon S3 コンソールで、*example-s3-bucket1* にアタッチされているバケットポリシーを削除します。バケットの [プロパティ] で、[アクセス許可] セクションのポリシーを削除します。
     - バケットをこの演習のために作成した場合は、Amazon S3 コンソールでオブジェクトを削除してから、バケットを削除します。

- IAM コンソールで、アカウント A で作成した `examplerole` を削除します。詳細な手順については、「IAM ユーザーガイド」の「IAM ユーザーの削除」を参照してください。
- IAM コンソールで、[AccountAadmin] ユーザーを削除します。

2. アカウント B の認証情報を使用して IAM コンソールにサインインします。ユーザー AccountBadmin を削除します。
3. アカウント C の認証情報を使用して IAM コンソールにサインインします。AccountCadmin とユーザー Dave を削除します。

関連リソース

このチュートリアルに関連する詳細については、「IAM ユーザーガイド」の以下のリソースを参照してください。

- IAM ユーザーにアクセス許可を委任するロールの作成
- チュートリアル: IAM ロールを使用した AWS アカウント 全体でのアクセスの委任
- IAM ポリシーを管理する

Amazon S3 がリクエストを許可する仕組み

Amazon S3 は、バケットまたはオブジェクトオペレーションなどのリクエストを受け取ると、まずリクエスタに必要なアクセス許可があることを確認します。Amazon S3 は、関連するすべてのアクセスポリシー、ユーザーポリシー、およびリソースベースのポリシー (バケットポリシー、バケットアクセスコントロールリスト (ACL)、オブジェクト ACL) を評価して、リクエストを許可するかどうかを決めます。

> **Note**
>
> Amazon S3 アクセス許可チェックで有効な許可が見つからなかった場合は、アクセス拒否 (403 Forbidden) アクセス許可拒否エラーが返されます。詳細については、「Amazon S3 でのアクセス拒否 (403 Forbidden) エラーのトラブルシューティング」を参照してください。

Amazon S3 は、リクエストを受け取ると、リクエスタが特定のオペレーションを実行するアクセス許可があるかどうかを確認するため、次の操作を順に実行します。

1. 関連するすべてのアクセスポリシー (ユーザーポリシー、バケットポリシー、および ACL) を実行時に評価ターゲットのポリシーのセットに変換します。
2. 作成したポリシーのセットを次の手順で評価します。各ステップで、Amazon S3 は、コンテキストの権限に基づいて、ポリシーのサブセットを特定のコンテキストで評価します。
   a. ユーザーコンテキスト – ユーザーコンテキストでは、ユーザーが属する親アカウントにコンテキストの権限があります。

      Amazon S3 は、親アカウントが所有するポリシーのサブセットを評価します。このサブセットには、親がユーザーにアタッチするユーザーポリシーが含まれています。親がリクエスト内のリソース (バケットまたはオブジェクト) も所有している場合、Amazon S3 は、対応するリソースポリシー (バケットポリシー、バケット ACL、オブジェクト ACL) も同時に評価します。

      ユーザーは、オペレーションを実行するための親アカウントからのアクセス許可を持つ必要があります。

      このステップは、リクエストが AWS アカウント内のユーザーによって行われた場合にのみ適用されます。リクエストが AWS アカウントのルートユーザー認証情報を使用して行われている場合、Amazon S3 はこのステップをスキップします。

b. バケットコンテキスト – バケットコンテキストでは、Amazon S3 はバケットを所有する AWS アカウント が所有するポリシーを評価します。

   バケットオペレーションに対するリクエストの場合、リクエスタはバケット所有者からのアクセス許可を持つ必要があります。リクエストがオブジェクトをターゲットにしている場合、Amazon S3 は、バケット所有者が所有するすべてのポリシーを評価して、バケット所有者がオブジェクトへのアクセスを明示的に拒否していないかどうかを確認します。明示的な拒否セットが存在する場合、Amazon S3 はリクエストを許可しません。

c. オブジェクトコンテキスト – オブジェクトに対するリクエストの場合、Amazon S3 はオブジェクトの所有者が所有するポリシーのサブセットを評価します。

Amazon S3 がリクエストを承認する方法を示すいくつかのシナリオの例を次に示します。

Example - リクエスタが IAM プリンシパルの場合

リクエスタが IAM プリンシパルの場合、Amazon S3 は、オペレーションを実行するために必要な許可が、プリンシパルが属する親 AWS アカウントよりプリンシパルに付与されているかどうかを判断する必要があります。さらに、リクエストがバケットオペレーション (バケット内容のリストを取得するリクエストなど) の場合、Amazon S3 は、オペレーションを実行するためのアクセス許可をバケット所有者がリクエスタに付与していることを確認する必要があります。IAM プリンシパルがリソースに対して特定のオペレーションを実行するには、そのプリンシパルが属する親 AWS アカウントと、リソースを所有する AWS アカウントの両方からの許可が必要です。

Example - リクエスタが IAM プリンシパルの場合 - リクエストが、バケット所有者が所有していないオブジェクトに対するオペレーションに関連している場合。

バケット所有者が所有していないオブジェクトに対するオペレーションのリクエストの場合は、リクエスタがオブジェクトの所有者からのアクセス許可を持つことの確認に加えて、Amazon S3 はバケットポリシーもチェックして、バケット所有者がオブジェクトに対する明示的な拒否を設定していないことを確認する必要があります。バケット所有者 (請求の支払者) は、オブジェクトの所有者にかかわらず、バケット内のオブジェクトへのアクセスを明示的に拒否できます。また、バケット所有者は、バケット内のすべてのオブジェクトを削除できます。

デフォルトでは、別の AWS アカウント がオブジェクトを S3 バケットにアップロードすると、そのアカウント (オブジェクトライター) がオブジェクトを所有し、そのオブジェクトにアクセスでき、アクセスコントロールリスト (ACL) を介して他のユーザーにそのオブジェクトへのアクセスを許可できます。オブジェクトの所有権を使用してこのデフォルトの動作を変更し、ACL を無効にして、

バケット所有者としてバケット内のすべてのオブジェクトを自動的に所有することができます。その結果、データのアクセスコントロールは、IAM ユーザーポリシー、S3 バケットポリシー、仮想プライベートクラウド (VPC) エンドポイントポリシー、AWS Organizations サービスコントロールポリシー (SCP) などのポリシーに基づいています。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

Amazon S3 がアクセスポリシーを評価してバケットオペレーションおよびオブジェクトオペレーションのリクエストを承認または拒否する仕組みについて詳しくは、以下のトピックを参照してください。

トピック

- Amazon S3 がバケットオペレーションのリクエストを承認する仕組み
- Amazon S3 がオブジェクトオペレーションのリクエストを許可する仕組み

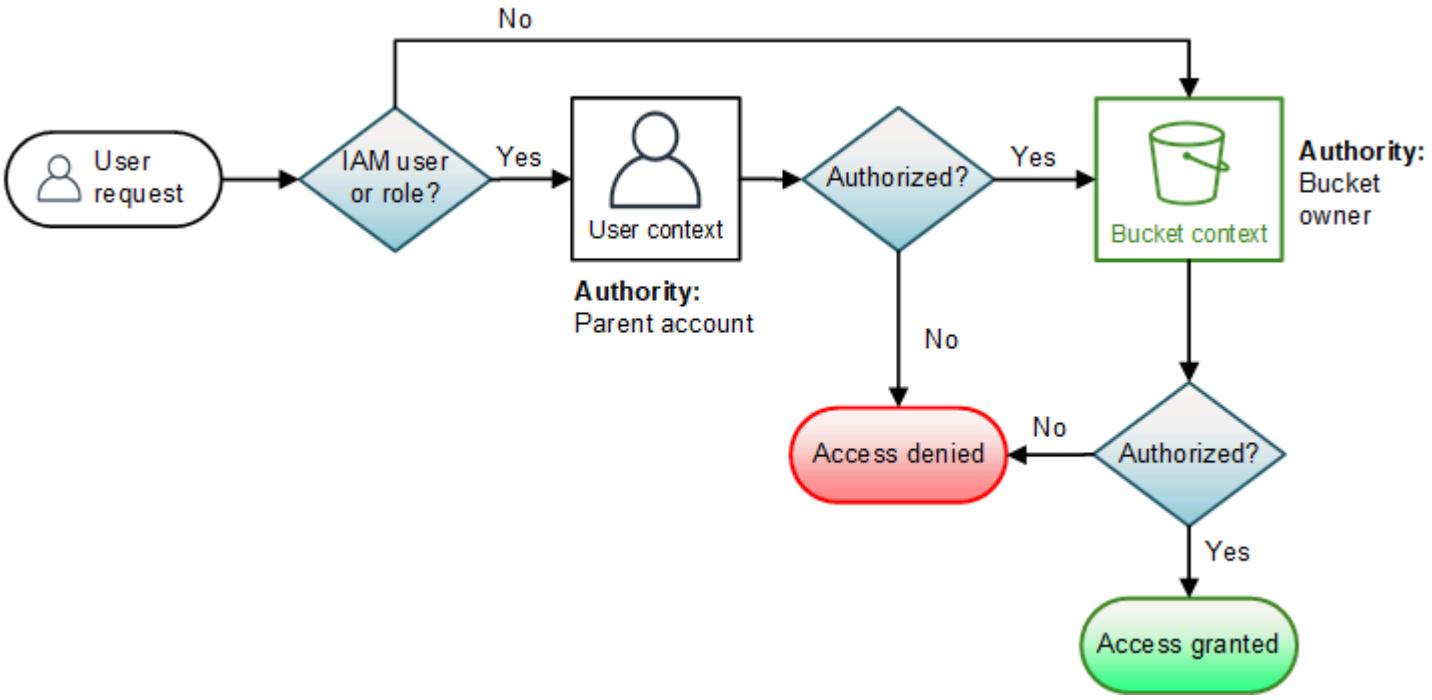
Amazon S3 がバケットオペレーションのリクエストを承認する仕組み

Amazon S3 は、バケットオペレーションのリクエストを受け取ると、関連するすべてのアクセス許可を、実行時に評価する一連のポリシーに変換します。関連するアクセス許可には、リソースベースのアクセス許可 (バケットポリシーやバケットアクセスコントロールリストなど) とユーザーポリシー (リクエストが IAM プリンシパルからの場合) が含まれます。Amazon S3 はその後、作成したポリシーのセットを特定のコンテキスト (ユーザーコンテキストまたはバケットコンテキスト) に従って、一連の手順で評価します。

1. ユーザーコンテキスト – リクエスタが IAM プリンシパルの場合、そのプリンシパルは親 AWS アカウント からアクセス許可を付与されている必要があります。このステップで、Amazon S3 は、親アカウント (コンテキストの権限とも呼ばれる) が所有するポリシーのサブセットを評価します。このポリシーのサブセットには、親アカウントがプリンシパルにアタッチするユーザーポリシーが含まれます。親がリクエスト内のリソース(この場合はバケット)も所有している場合、Amazon S3 は、対応するリソースポリシー(バケットポリシーおよびバケット ACL)も同時に評価します。バケットオペレーションのリクエストが出されるたびに、サーバーアクセスログはリクエスタの正規 ID を記録します。詳細については、「サーバーアクセスログによるリクエストの口グ記録」を参照してください。
2. バケットコンテキスト – リクエスタは、バケット所有者から特定のバケットオペレーションを実行するためのアクセス許可を付与されている必要があります。このステップで、Amazon S3 は、バケットを所有する AWS アカウントが所有するポリシーのサブセットを評価します。

バケット所有者は、バケットポリシーまたはバケット ACL を使用してアクセス許可を付与できます。バケットを所有している AWS アカウント が IAM プリンシパルの親アカウントでもある場合は、ユーザーポリシーでバケットの許可を設定できます。

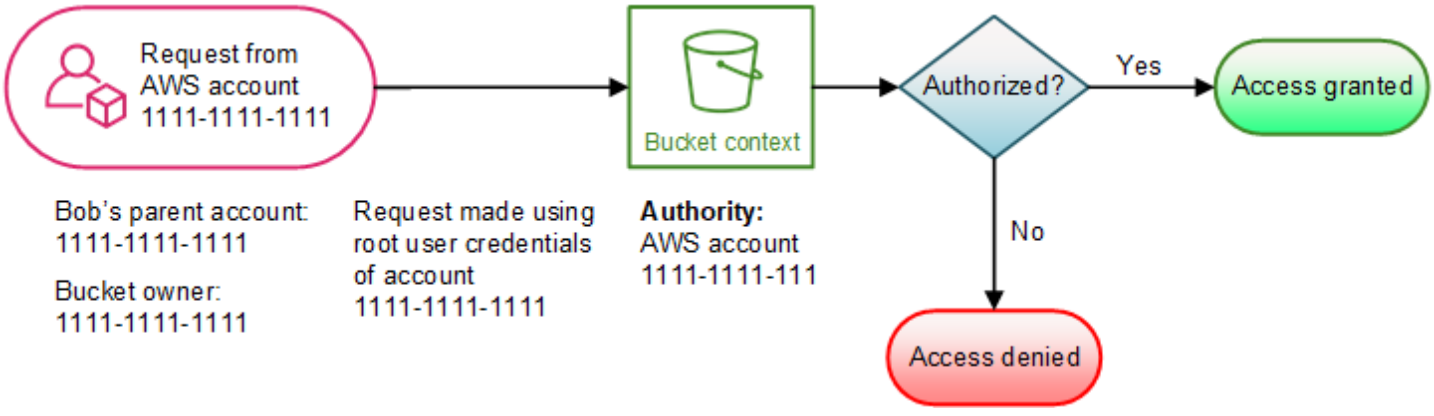
次に示すのは、バケットオペレーションのコンテキストベースの評価を説明するための図です。



次の例は、評価ロジックを示します。

例 1: バケット所有者がリクエストするバケットオペレーション

この例では、バケット所有者が AWS アカウントのルート認証情報を使用してバケットオペレーションへのリクエストを送信します。

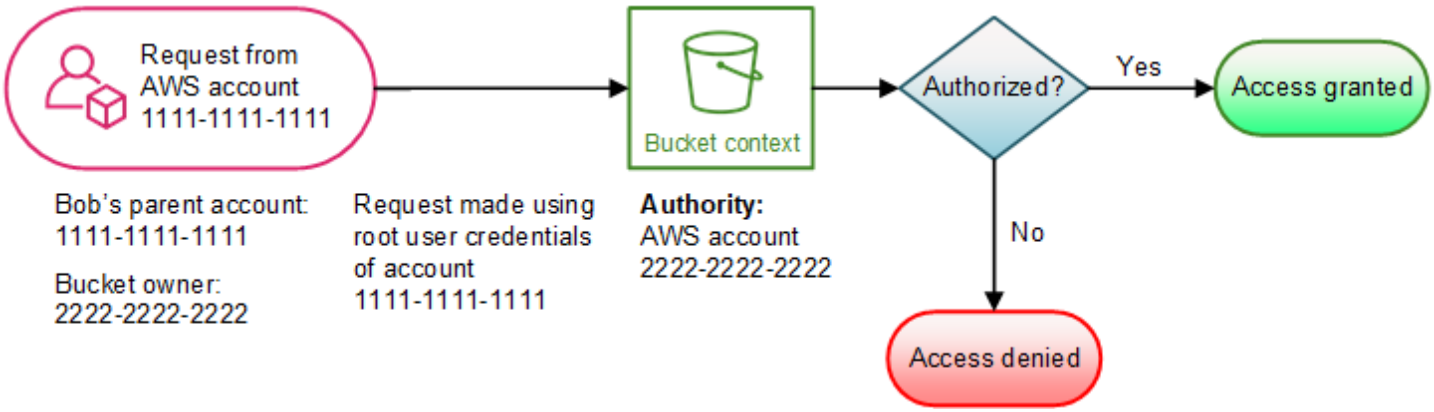


Amazon S3 はコンテキストの評価を次のように実行します。

1. リクエストは AWS アカウントのルートユーザー認証情報を使用して行われるため、ユーザーコンテキストは評価されません。
2. バケットコンテキストでは、Amazon S3 はバケットポリシーを調べて、リクエスタがオペレーションを実行するアクセス許可を持つかどうかを判定します。Amazon S3 はリクエストを許可します。

例 2: バケット所有者でない AWS アカウントがリクエストしたバケットオペレーション

この例では、AWS アカウント 2222-2222-2222 が所有するバケットオペレーションで AWS アカウント 1111-1111-1111 のルートユーザー認証情報を使用してリクエストが行われます。このリクエストに IAM ユーザーは関与していません。

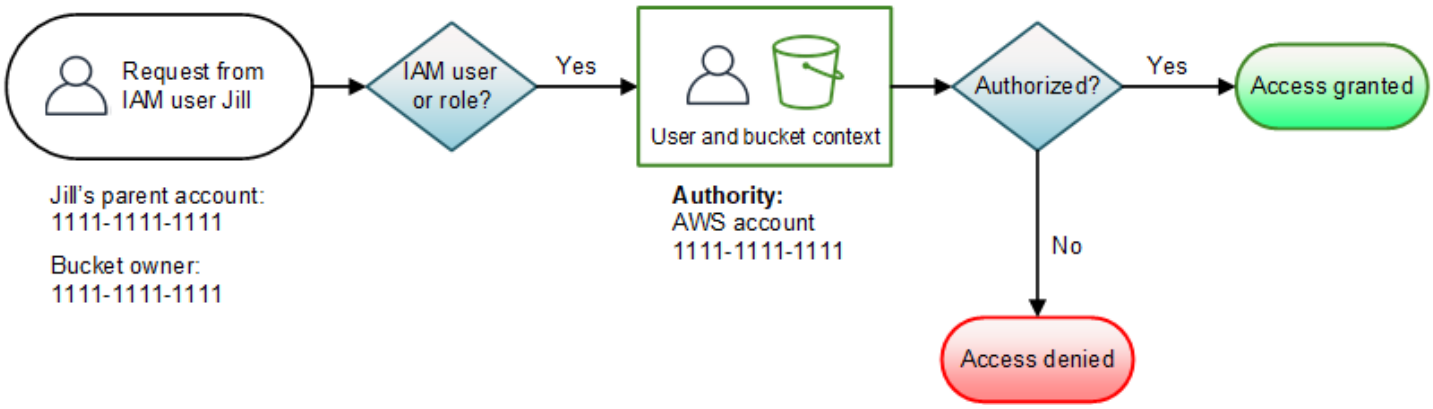


この例では、Amazon S3 は次のようにコンテキストを評価します。

1. リクエストは AWS アカウントのルートユーザー認証情報を使用して行われるため、ユーザーコンテキストは評価されません。
2. バケットコンテキストでは、Amazon S3 はバケットポリシーを調べます。バケット所有者 (AWS アカウント 2222-2222-2222) が、リクエストされたオペレーションを AWS アカウント 1111-1111-1111 に許可していない場合は、Amazon S3 はリクエストを拒否します。それ以外の場合、Amazon S3 はリクエストを許可し、オペレーションを実行します。

例 3: 親 AWS アカウントがバケット所有者でもある IAM プリンシパルがリクエストしたバケットオペレーション

この例では、リクエストを送信したユーザーが AWS アカウント 1111-1111-1111 の IAM ユーザー Jill であり、このアカウントがバケット所有者でもあります。



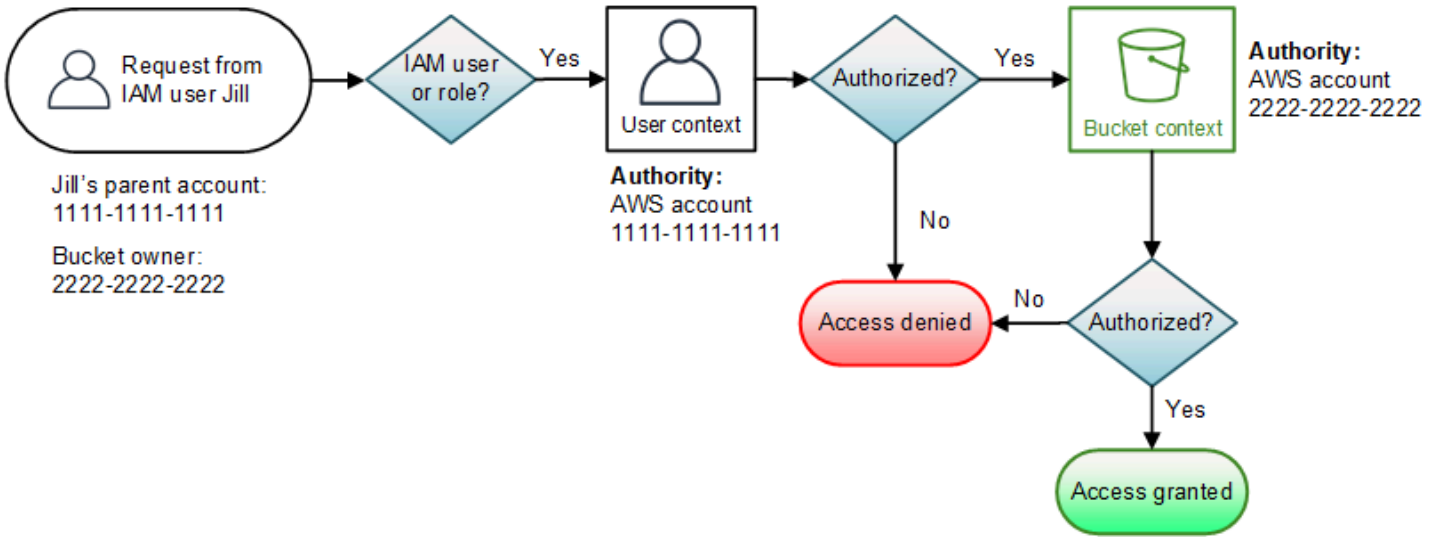
Amazon S3 は次のコンテキスト評価を実行します。

1. リクエストは IAM プリンシパルからであるため、ユーザーコンテキストで、Amazon S3 は親 AWS アカウントに属するすべてのポリシーを評価して、Jill がオペレーションを実行する許可を持っているかどうかを判定します。

   この例では、プリンシパルが属する親 AWS アカウント 1111-1111-1111 は、バケット所有者でもあります。その結果、Amazon S3 は、ユーザーポリシーに加えて、バケットポリシーとバケット ACL も同じコンテキストで評価します。これらが同じアカウントに属するからです。
2. Amazon S3 はバケットポリシーとバケット ACL をユーザーコンテキストの一部として評価したので、バケットコンテキストは評価しません。

例 4: 親 AWS アカウントがバケット所有者でない IAM プリンシパルがリクエストしたバケットオペレーション

この例では、リクエストを送信したのが 1111-1111-1111 を親 AWS アカウント とする IAM ユーザー Jill であり、バケット所有者は別の AWS アカウント 2222-2222-2222 です。



Jill には、親 AWS アカウントとバケット所有者の両方からの許可が必要です。Amazon S3 は次のようにコンテキストを評価します。

1. リクエストは IAM プリンシパルからであるため、Amazon S3 は、アカウントが作成したポリシーを参照してユーザーコンテキストを評価し、Jill に必要なアクセス許可があることを検証します。Jill にアクセス許可がある場合、Amazon S3 はバケットコンテキストの評価に進みます。Jill にアクセス許可がない場合、Amazon S3 はリクエストを拒否します。
2. バケットコンテキストでは、Amazon S3 はバケット所有者の 2222−2222−2222 がリクエストされたオペレーションを実行するアクセス許可を Jill (またはその親 AWS アカウント) に付与していることを確認します。アクセス許可がある場合、Amazon S3 はリクエストを許可し、オペレーションを実行します。アクセス許可がない場合、Amazon S3 はリクエストを拒否します。

Amazon S3 がオブジェクトオペレーションのリクエストを許可する仕組み

Amazon S3 は、オブジェクトオペレーションのリクエストを受け取ると、関連するすべての許可 、リソースベースの許可 (オブジェクトアクセスコントロールリスト (ACL)、バケットポリシー、バケット (ACL) と IAM ユーザーポリシー) を実行時に評価する一連のポリシーに変換します。その後、作成したポリシーのセットを一連の手順で評価します。各ステップでは、ポリシーのサブセットが、ユーザーコンテキスト、バケットコンテキスト、オブジェクトコンテキストの 3 つの固有のコンテキストで評価されます。

1. ユーザーコンテキスト – リクエスタが IAM プリンシパルの場合、そのプリンシパルは親 AWS アカウント からアクセス許可を付与されている必要があります。このステップで、Amazon S3 は、親アカウント (コンテキストの権限とも呼ばれる) が所有するポリシーのサブセットを評価します。このポリシーのサブセットには、親がプリンシパルにアタッチするユーザーポリシー

が含まれます。親がリクエスト内のリソース (バケットまたはオブジェクト) も所有している場合、Amazon S3 は、対応するリソースポリシー (バケットポリシー、バケット ACL、オブジェクト ACL) も同時に評価します。

> **Note**
>
> 親 AWS アカウントがリソース (バケットまたはオブジェクト) を所有している場合、親アカウントは、ユーザーポリシーまたはリソースポリシーを使用して、リソースの許可を IAM プリンシパルに付与できます。

2. バケットコンテキスト – このコンテキストでは、Amazon S3 はバケットを所有する AWS アカウント が所有するポリシーを評価します。

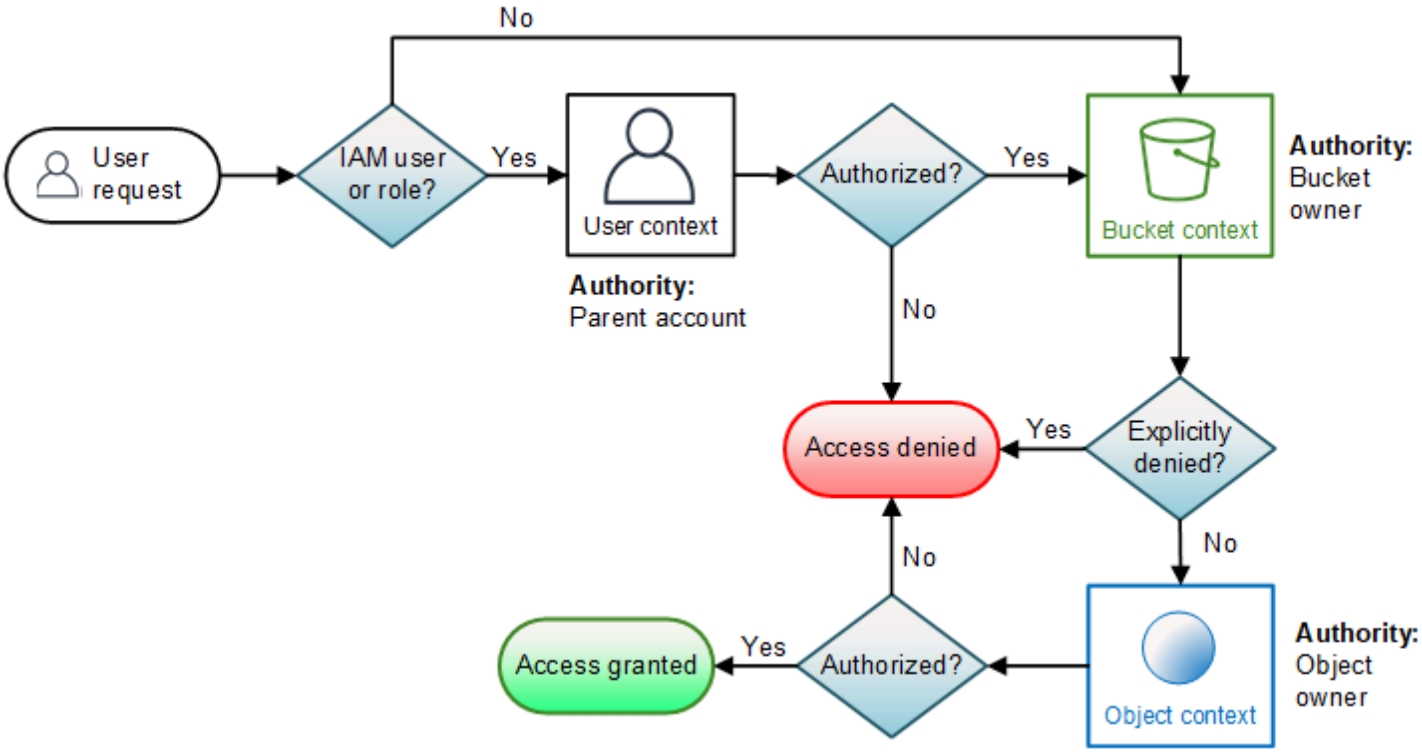
   リクエスト内のオブジェクトを所有する AWS アカウント がバケット所有者と同じでない場合、Amazon S3 はポリシーをチェックして、バケット所有者がオブジェクトへのアクセスを明示的に拒否しているかどうかをチェックします。オブジェクトに対する明示的な拒否セットが存在する場合、Amazon S3 はリクエストを許可しません。

3. オブジェクトコンテキスト – リクエスタは、オブジェクト所有者から特定のオブジェクトオペレーションを実行するためのアクセス許可を付与されている必要があります。このステップで、Amazon S3 はオブジェクト ACL を評価します。

> **Note**
>
> バケットとオブジェクトの所有者が同じ場合、オブジェクトへのアクセスは、バケットコンテキストで評価されるバケットポリシーで許可することができます。所有者が異なる場合、オブジェクトの所有者はオブジェクト ACL を使用してアクセス許可を付与する必要があります。オブジェクトを所有する AWS アカウントが IAM プリンシパルの親アカウントでもある場合は、ユーザーコンテキストで評価されるユーザーポリシーでオブジェクトのアクセス許可を設定できます。これらのアクセスポリシーの各オプションの詳細については、チュートリアル: ポリシーを使用した Amazon S3 リソースへのアクセスの管理 を参照してください。
>
> バケット所有者がバケット内のすべてのオブジェクトを所有し、バケットポリシーまたは IAM ベースのポリシーを使用してこれらのオブジェクトへのアクセスを管理する場合は、オブジェクト所有権にバケット所有者強制設定を適用できます。この設定では、バケット所有者として、バケット内のすべてのオブジェクトを自動的に所有し、完全に制御できます。バケットとオブジェクト ACL は編集できず、アクセスと見なされなくなります。詳

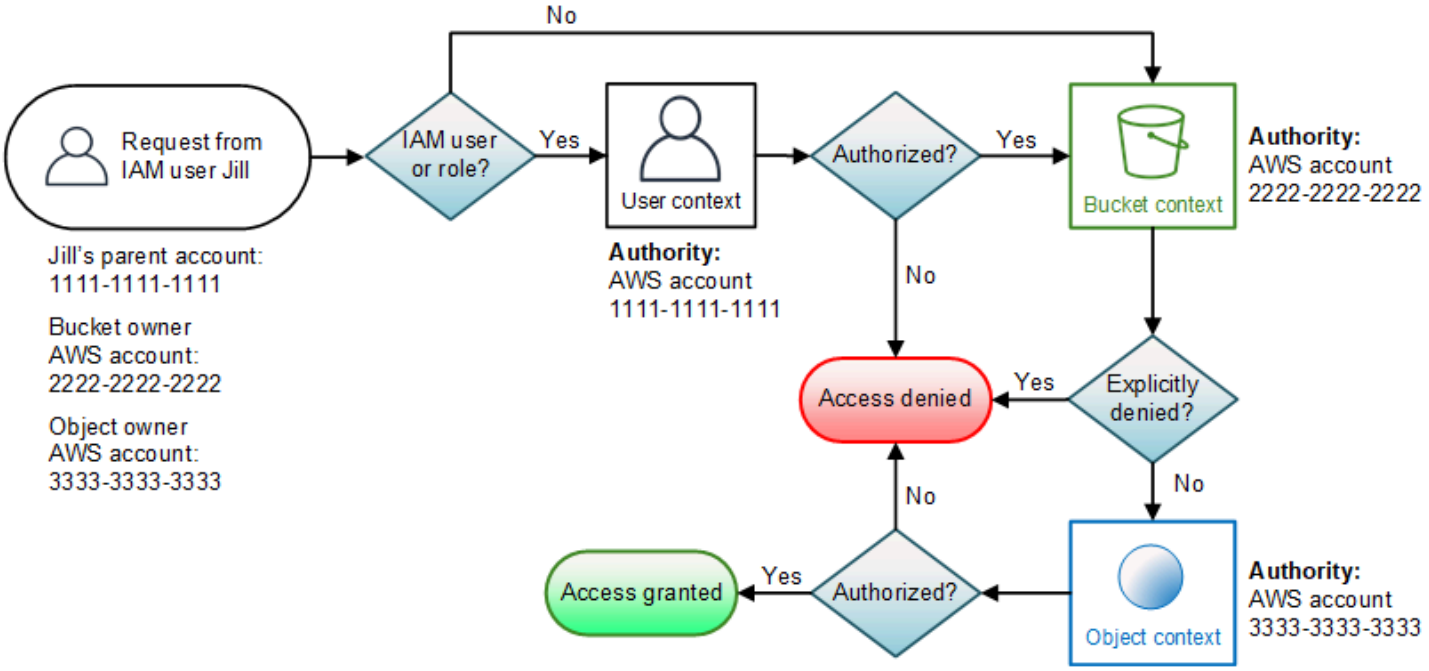
細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

次に示すのは、オブジェクトオペレーションに対するコンテキストベースの評価の説明です。



オブジェクトオペレーションのリクエストの例

この例では、親 AWS アカウント が 1111-1111-1111 である IAM ユーザー Jill が、GetObject 2222-2222-2222 が所有するバケットにある、AWS アカウント 3333-3333-3333 が所有するオブジェクトに対するオブジェクトオペレーションのリクエスト (AWS アカウント など) を送信します。



Jill には、親 AWS アカウント、バケット所有者、およびオブジェクト所有者からの許可が必要です。Amazon S3 は次のようにコンテキストを評価します。

1. リクエストは IAM プリンシパルからであるため、Amazon S3 は、ユーザーコンテキストを評価して、親 AWS アカウント 1111-1111-1111 がリクエストされたオペレーションを実行するアクセス許可を Jill に付与していることを検証します。アクセス許可がある場合、Amazon S3 はバケットコンテキストを評価します。アクセス許可がない場合、Amazon S3 はリクエストを拒否します。
2. バケットコンテキストでは、バケット所有者の AWS アカウント 2222-2222-2222 がコンテキストの権限です。Amazon S3 は、バケットポリシーを評価して、バケット所有者がオブジェクトへのアクセスを Jill に対して明示的に拒否しているかどうかを判定します。
3. オブジェクトコンテキストでは、コンテキストの権限はオブジェクト所有者の AWS アカウント 3333-3333-3333 です。Amazon S3 は、オブジェクト ACL を評価して、Jill がオブジェクトにアクセスするアクセス許可を持つかどうかを判定します。アクセス許可がある場合、Amazon S3 はリクエストを許可します。

Amazon S3 の AWS マネージドポリシー

AWS マネージドポリシーは、AWS が作成および管理するスタンドアロンポリシーです。AWS マネージドポリシーは、多くの一般的ユースケースでアクセス許可を提供するように設計されているため、ユーザー、グループ、ロールへのアクセス許可の割り当てを開始できます。

AWS マネージドポリシーは、ご利用の特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることにご注意ください。AWSのすべてのお客様が使用できるようになるのを避けるためです。ユースケース別にカスタマーマネージドポリシーを定義して、マネージドポリシーを絞り込むことをお勧めします。

AWS マネージドポリシーで定義したアクセス権限は変更できません。AWS が AWS マネージドポリシーに定義されている権限を更新すると、更新はポリシーがアタッチされているすべてのプリンシパルアイデンティティ (ユーザー、グループ、ロール) に影響します。新しい AWS のサービス を起動するか、既存のサービスで新しい API オペレーションが使用可能になると、AWS が AWS マネージドポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS マネージドポリシー: AmazonS3FullAccess

`AmazonS3FullAccess` ポリシーは IAM アイデンティティにアタッチできます。このポリシーは、Amazon S3 への完全なアクセスを可能にする許可を付与します。

このポリシーのアクセス許可を確認するには、「AWS Management Console」の「AmazonS3FullAccess」を参照してください。

AWS マネージドポリシー: AmazonS3ReadOnlyAccess

`AmazonS3ReadOnlyAccess` ポリシーは IAM アイデンティティにアタッチできます。このポリシーは、Amazon S3 への読み取り専用アクセスを可能にする許可を付与します。

このポリシーのアクセス許可を確認するには、「AWS Management Console」の「AmazonS3ReadOnlyAccess」を参照してください。

AWS マネージドポリシー: AmazonS3ObjectLambdaExecutionRolePolicy

S3 Object Lambda アクセスポイントにリクエストが行われたときに S3 Object Lambda にデータを送信するために必要な許可を AWS Lambda 関数に提供します。また、Amazon CloudWatch Logs に書き込む許可を Lambda に付与します。

このポリシーのアクセス許可を確認するには、「AWS Management Console」の「AmazonS3ObjectLambdaExecutionRolePolicy」を参照してください。

AWS マネージドポリシーに対する Amazon S3 更新

Amazon S3 の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。

変更	説明	日付
Amazon S3 が AmazonS3ReadOnlyAccess に記述許可を追加しました	Amazon S3 が s3:Describe* 記述許可を AmazonS3ReadOnlyAccess に追加しました。	2023 年 8 月 11 日
Amazon S3 では、S3 Object Lambda のアクセス許可を AmazonS3FullAccess および AmazonS3ReadOnlyAccess に追加しました。	Amazon S3 は、S3 Object Lambda のアクセス権限を含む AmazonS3FullAccess および AmazonS3ReadOnlyAccess のポリシーを更新しました。	2021 年 9 月 27 日
Amazon S3 が AmazonS3ObjectLambdaExecutionRolePolicy を追加	Amazon S3 は、AmazonS3ObjectLambdaExecutionRolePolicy と呼ばれる新しい AWS マネージドポリシーを追加しました。これは、Lambda 関数に S3 Object Lambda とやり取りして CloudWatch Logs に書き込むための許可を提供します。	2021 年 8 月 18 日
Amazon S3 が変更の追跡を開始	Amazon S3 が AWS マネージドポリシーの変更の追跡を開始しました。	2021 年 8 月 18 日

Amazon S3 ストレージレンズでのサービスにリンクされたロールの使用

Amazon S3 ストレージレンズを使用して AWS Organizations 内のすべてのアカウントでメトリクスを収集および集約するには、最初に組織の管理アカウントにより有効化された、信頼されたアクセスを S3 Storage Lens が持っていることを確認する必要があります。S3 Storage Lens は、サービスリンクロール (SLR) を作成し、そのロールが組織に属する AWS アカウント のリストを取得できるようにします。このアカウントの一覧は、S3 Storage Lens ダッシュボードまたは設定が作成または更新された場合に、すべてのメンバーアカウントの S3 リソースのメトリクスを収集するため S3 Storage Lens によって使用されます。

Amazon S3 ストレージレンズは AWS Identity and Access Management (IAM) サービスにリンクされたロールを使用します。サービスにリンクされたロールは S3 Storage Lens に直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、S3 Storage Lens によって事前定義されており、ユーザーの代わりにサービスから他の AWS のサービス を呼び出す必要のある許可がすべて含まれています。

サービスにリンクされたロールを使用すると、必要なアクセス許可を手動で追加する必要がなくなるため、S3 Storage Lens の設定が簡単になります。S3 Storage Lens は、サービスにリンクされたロールのアクセス許可を定義します。特に定義されている場合を除き、S3 Storage Lens のみがそのロールを引き受けることができます。定義された許可には信頼ポリシーと許可ポリシーが含まれ、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

関連リソースを削除した後でなければ、このサービスにリンクされたロールを削除することはできません。これにより、リソースにアクセスするためのアクセス許可を不注意で削除することが防止され、S3 Storage Lens リソースは保護されます。

サービスにリンクされたロールをサポートするその他のサービスについては、「IAM と連携する AWS サービス」を参照の上、サービスにリンクされたロール列がはいになっているサービスを検索してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Amazon S3 ストレージレンズへのサービスにリンクされたロールのアクセス許可

S3 Storage Lens は、AWSServiceRoleForS3StorageLens という名前のサービスにリンクされたロールを使用します。これにより、S3 Storage Lens によって使用または管理される AWS のサービスとリソースにアクセスできます。これにより、S3 Storage Lens は、ユーザーに代わって AWS Organizations のリソースにアクセスできるようになります。

S3 Storage Lens のサービスにリンクされたロールは、組織のストレージ上で次のサービスを信頼します。

- `storage-lens.s3.amazonaws.com`

ロールのアクセス許可ポリシーは、以下のアクションを実行することを S3 Storage Lens に許可します。

- `organizations:DescribeOrganization`

  `organizations:ListAccounts`

```
organizations:ListAWSServiceAccessForOrganization

organizations:ListDelegatedAdministrators
```

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、アクセス許可を設定する必要があります。詳細については、IAM ユーザーガイドのサービスにリンクされたロールのアクセス許可を参照してください。

S3 Storage Lens へのサービスにリンクされたロールの作成

サービスにリンクされたロールを手動で作成する必要はありません。AWS Organizations 管理アカウントまたは委任された管理者アカウントにサインインしているときに次のいずれかのタスクを完了すると、S3 Storage Lens がサービスにリンクされたロールを作成します。

- Amazon S3 コンソールで、組織用の S3 Storage Lens ダッシュボード設定を作成する。
- REST API、AWS CLI、SDK を使用して組織に S3 Storage Lens 設定を PUT する。

> **Note**
>
> S3 Storage Lens は組織ごとに最大 5 人の委任管理者をサポートします。

このサービスにリンクされたロールを削除した場合、前述のアクションによって必要に応じてロールが再作成されます。

S3 Storage Lens サービスにリンクされたロールのポリシーの例

Example S3 Storage Lens のサービスにリンクされたロールへのアクセス許可ポリシー

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AwsOrgsAccess",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAccounts",
                "organizations:ListAWSServiceAccessForOrganization",
```

```
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Amazon S3 ストレージレンズのサービスにリンクされたロールの編集

S3 Storage Lens では、AWSServiceRoleForS3StorageLens サービスリンクロールを編集することはできません。サービスリンクロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロール記述の編集はできます。詳細については、IAM ユーザーガイドのサービスにリンクされたロールの編集を参照してください。

Amazon S3 ストレージレンズのサービスにリンクされたロールの削除

サービスにリンクされたロールが不要になった場合は、そのロールを削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニタリングされたり、メンテナンスされたりすることがなくなります。ただし、手動で削除する前に、サービスにリンクされたロールのリソースをクリーンアップする必要があります。

> **Note**
>
> リソースを削除する際に、Amazon S3 ストレージレンズサービスでそのロールが使用されている場合、削除は失敗することがあります。失敗した場合は、数分待ってから再度オペレーションを実行してください。

AWSServiceRoleForS3StorageLens を削除するには、AWS Organizations 管理アカウントまたは委任された管理者アカウントを使用して、すべての AWS リージョン に存在する組織レベルの S3 Storage Lens 設定をすべて削除する必要があります。

リソースは組織レベルの S3 Storage Lens 設定です。S3 Storage Lens を使用してリソースをクリーンアップし、IAM コンソール、CLI、REST API、または AWS SDK を使用してロールを削除します。

REST API、AWS CLI、SDK では、`ListStorageLensConfigurations` を使用して組織が S3 Storage Lens 設定を作成したすべてのリージョンで S3 Storage Lens 設定を見つけることができま

す。DeleteStorageLensConfiguration アクションを使用してこれらの設定を削除し、ロールを削除できるようにします。

> **Note**
>
> サービスにリンクされたロールを削除するには、すべてのリージョンに存在する組織レベルの S3 Storage Lens 設定をすべて削除する必要があります。

AWSServiceRoleForS3StorageLens SLR で使用されている Amazon S3 ストレージレンズリソースを削除するには

1. 組織レベルの設定一覧を取得するには、S3 Storage Lens 設定があるすべてのリージョンで ListStorageLensConfigurations を使用する必要があります。この一覧は Amazon S3 コンソールから取得することもできます。
2. これらの設定は、DeleteStorageLensConfiguration API コールを呼び出すか Amazon S3 コンソールを使用して、適切なリージョンのエンドポイントから削除する必要があります。

サービスにリンクされたロールを IAM で手動削除するには

設定を削除した後、IAM コンソールから、または IAM API DeleteServiceLinkedRole を呼び出すか、AWS CLI もしくは AWS SDK を使用して、AWSServiceRoleForS3StorageLens SLR を削除できます。詳細については、IAM ユーザーガイド のサービスにリンクされたロールの削除を参照してください。

S3 Storage Lens のサービスにリンクされたロールがサポートされるリージョン

S3 Storage Lens は、そのサービスを利用できるすべての AWS リージョンで、サービスにリンクされたロールの使用をサポートします。詳細については、Amazon S3 Regions and Endpoints を参照してください。

Amazon S3 アイデンティティとアクセスのトラブルシューティング

以下の情報を使用して、Amazon S3 と IAM の使用時に発生する可能性がある一般的な問題の診断と修正に役立てます。

トピック

- アクセス拒否エラーが表示された
- Amazon S3 でアクションを実行する権限がありません

- iam:PassRole を実行する権限がありません
- 自分の AWS アカウント 以外のユーザーに Amazon S3 リソースへのアクセスを許可したい

アクセス拒否エラーが表示された

バケットポリシーまたはアイデンティティベースポリシーのいずれにも、アクセス許可を付与するリクエスタに対する明示的な Deny ステートメントがないことを確認します。

アクセス拒否エラーのトラブルシューティングの詳細については、「Amazon S3 でのアクセス拒否 (403 Forbidden) エラーのトラブルシューティング」を参照してください。

Amazon S3 でアクションを実行する権限がありません

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要な s3:*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 s3:GetWidget on resource: my-example-widget
```

この場合、s3:*GetWidget* アクションを使用して *my-example-widget* リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して Amazon S3 にロールを渡せるようにする必要があります。

一部の AWS のサービスでは、新しいサービスロールやサービスリンクロールを作成せずに、既存のロールをサービスに渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

次のエラー例は、marymajor という IAM ユーザーがコンソールを使用して Amazon S3 でアクションを実行しようとする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。サインイン認証情報を提供した担当者が管理者です。

自分の AWS アカウント 以外のユーザーに Amazon S3 リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください。

- Amazon S3 がこれらの機能をサポートしているかどうかを確認するには、「Amazon S3 での IAM の機能」を参照してください。
- 所有している AWS アカウント 全体のリソースへのアクセス権を提供する方法については、IAM ユーザーガイドの所有している別の AWS アカウント アカウントへのアクセス権を IAM ユーザーに提供を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「サードパーティが所有する AWS アカウント へのアクセス権を付与する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイドの外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

S3 Access Grants でのアクセス管理

最小特権の原則に従うには、アプリケーション、ペルソナ、グループ、または組織単位に基づいて Amazon S3 データへのきめ細かなアクセスを定義します。アクセスパターンの規模と複雑さに応じて、さまざまなアプローチを使用して Amazon S3 のデータへのきめ細いアクセスを実現できます。

Amazon S3 内の少数から中程度の数のデータセットへのアクセスを AWS Identity and Access Management (IAM) プリンシパルで管理する最も簡単な方法は、IAM アクセス許可ポリシー と S3 バケットポリシー を定義することです。この戦略は、必要なポリシーが S3 バケットポリシー (20 KB) と IAM ポリシー (5 KB) のポリシーサイズ制限内に収まり、アカウントごとに許可されている IAM プリンシパルの数内である限り有効です。

データセットの数やユースケースの数が増えるにつれて、より多くのポリシースペースが必要になる場合があります。ポリシーステートメントの容量を大幅に増やすには、S3 バケットの追加エンドポイントとして S3 アクセスポイントを使用する方法があります。これは、各アクセスポイントに独自のポリシーを設定できるためです。アカウントにつき AWS リージョン ごとに数千のアクセス ポイントを設定でき、アクセスポイントごとに最大 20 KB のサイズのポリシーを設定できるため、非常にきめ細かいアクセスコントロール パターンを定義できます。S3 Access Points を使用すると、使用可能なポリシースペースの量が増えますが、クライアントが適切なデータセットに適したアクセスポイントを検出するためのメカニズムが必要になります。

3 番目の方法は、IAM セッションブローカーパターンの実装です。このパターンでは、アクセス決定ロジックを実装し、アクセスセッションごとに短期の IAM セッション認証情報を動的に生成します。IAM セッションブローカーの方法の場合、任意の動的なアクセス許可パターンをサポートして、効果的にスケールできる一方、アクセスパターンロジックを構築する必要があります。

このようなアプローチを使用する代わりに、S3 Access Grants を使用して、Amazon S3 データへのアクセスを管理できます。S3 Access Grants は、Amazon S3 内のデータへのアクセス許可をプレフィックス、バケット、またはオブジェクトごとに定義するための簡略化されたモデルを提供します。さらに、S3 Access Grants を使用して IAM プリンシパルにアクセスを付与することも、社内ディレクトリのユーザーまたはグループに直接アクセスを付与することもできます。

Amazon S3 のデータへのアクセス許可は、通常、ユーザーとグループをデータセットにマップすることで定義します。S3 Access Grants を使用して、Amazon S3 バケットとオブジェクト内のユーザーとロールへの S3 プレフィックスごとに直接アクセスマッピングを定義できます。S3 Access Grants のシンプルなアクセススキームを使用すると、読み取り専用、書き込み専用、または読み取り/書き込みのアクセス権限を S3 プレフィックスごとに IAM プリンシパルに付与することも、社内ディレクトリのユーザーまたはグループに直接付与することもできます。このような S3 Access Grants 機能を使用すると、アプリケーションはアプリケーションの現在の認証済みユーザーに代わって Amazon S3 にデータをリクエストできます。

S3 Access Grants を AWS IAM Identity Center の信頼されたアイデンティティプロパゲーション機能と統合すると、アプリケーションは、認証済みの社内ディレクトリユーザーに代わって (S3 Access Grants を含む) AWS のサービス に直接リクエストすることができます。アプリケーションは最初にユーザーを IAM プリンシパルにマッピングする必要がなくなります。さらに、エンドユーザーのア

イデンティティが Amazon S3 にまでプロパゲートされるため、どのユーザーがどの S3 オブジェクトにアクセスしたかの監査が簡素化されます。さまざまなユーザーと IAM セッション間の関係性を再構築する必要がなくなりました。S3 Access Grants と IAM アイデンティティセンターの信頼されたアイデンティティプロパゲーションを連携すると、Amazon S3 の各 AWS CloudTrail データイベントにデータアクセスがどのエンドユーザーの代わりなのかについての直接のリファレンスが含まれます。

S3 Access Grants の詳細については、後続のトピックを参照してください。

トピック

S3 Access Grants の概念

S3 Access Grants では、シンプルなアクセススキームとして以下の概念が導入されています。

S3 Access Grants インスタンス

S3 Access Grants インスタンスは、Amazon S3 データに対してどのユーザーがどのレベルのアクセスを持つかを定義する個別のアクセス許可の論理コンテナです。AWS リージョン ごと、AWS アカウント ごとに単一の S3 Access Grants インスタンスを持つことができます。この S3 Access Grants インスタンスを使用して、同じアカウントと AWS リージョン のすべてのバケットへのアクセスを制御できます。S3 Access Grants を使用して社内ディレクトリ内のユーザーとグループのアイデンティティへのアクセスを許可する場合は、S3 Access Grants インスタ

ンスを AWS Identity and Access Management (IAM) アイデンティティセンターのインスタンスに関連付ける必要もあります。

ロケーション

ロケーションは、S3 Access Grants インスタンスがアクセスを許可できるデータを定義します。S3 Access Grants は、特定の S3 プレフィックス、バケット、またはオブジェクトへのアクセスを範囲とする IAM 認証情報を発行することで機能します。S3 Access Grants のロケーションを IAM ロールに関連付けることで、そのロールから一時的なセッションが作成されます。最も一般的なロケーション設定は、S3 Access Grants インスタンス全体を `s3://` の単一のロケーションに設定することです。これで、アカウントと AWS リージョン 内のすべての S3 バケットへのアクセスに対応できます。S3 Access Grants インスタンスでは複数のロケーションを作成することもできます。例えば、バケットをこのバケットに制限する権限のために `s3://example-s3-bucket1` ロケーションとして登録できます。また、デフォルトのロケーション `s3://` も登録できます。

権限

ロケーション内のアクセス範囲を絞り込むには、個別の権限を作成します。S3 Access Grants インスタンスの個別の権限を使用すると、特定のエンティティ (IAM プリンシパル、または社内ディレクトリのユーザーやグループ) に Amazon S3 プレフィックス、バケット、またはオブジェクトへのアクセスを許可できます。権限ごとに、異なる範囲 (プレフィックス、バケット、またはオブジェクト) とアクセスレベル (`READ`、`WRITE`、または `READWRITE`) を定義できます。例えば、特定の社内ディレクトリグループ `01234567-89ab-cdef-0123-456789abcdef` に `s3://example-s3-bucket1/projects/items/*` への `READ` アクセスを許可する権限があるとします。この権限により、そのグループのユーザーは、*example-s3-bucket1* という名前のバケット内のプレフィックス `projects/items/` が付いたキー名を持つすべてのオブジェクトに `READ` アクセスが付与されます。

S3 Access Grants の一時的な認証情報

アプリケーションは、新しい S3 API オペレーション GetDataAccess を呼び出して、アクセス許可レベルが、`READ`、`WRITE` または `READWRITE` の単一のオブジェクト、プレフィックス、またはバケットへのアクセスをリクエストすることで、ジャストインタイムアクセス認証情報をリクエストできます。S3 Access Grants インスタンスは、`GetDataAccess` リクエストを保持する権限に照会して評価します。一致する権限がある場合、S3 Access Grants は、一致する権限のロケーションに関連付けられた IAM ロールを引き受けます。次に、S3 Access Grants は IAM セッションのアクセス許可の範囲を、権限の範囲で指定されている S3 バケット、プレフィックス、またはオブジェクトのみに限定します。一時アクセス認証情報の有効期限はデフォルトで 1 時間です。ただし、15 分から 12 時間までの任意の値に設定できます。

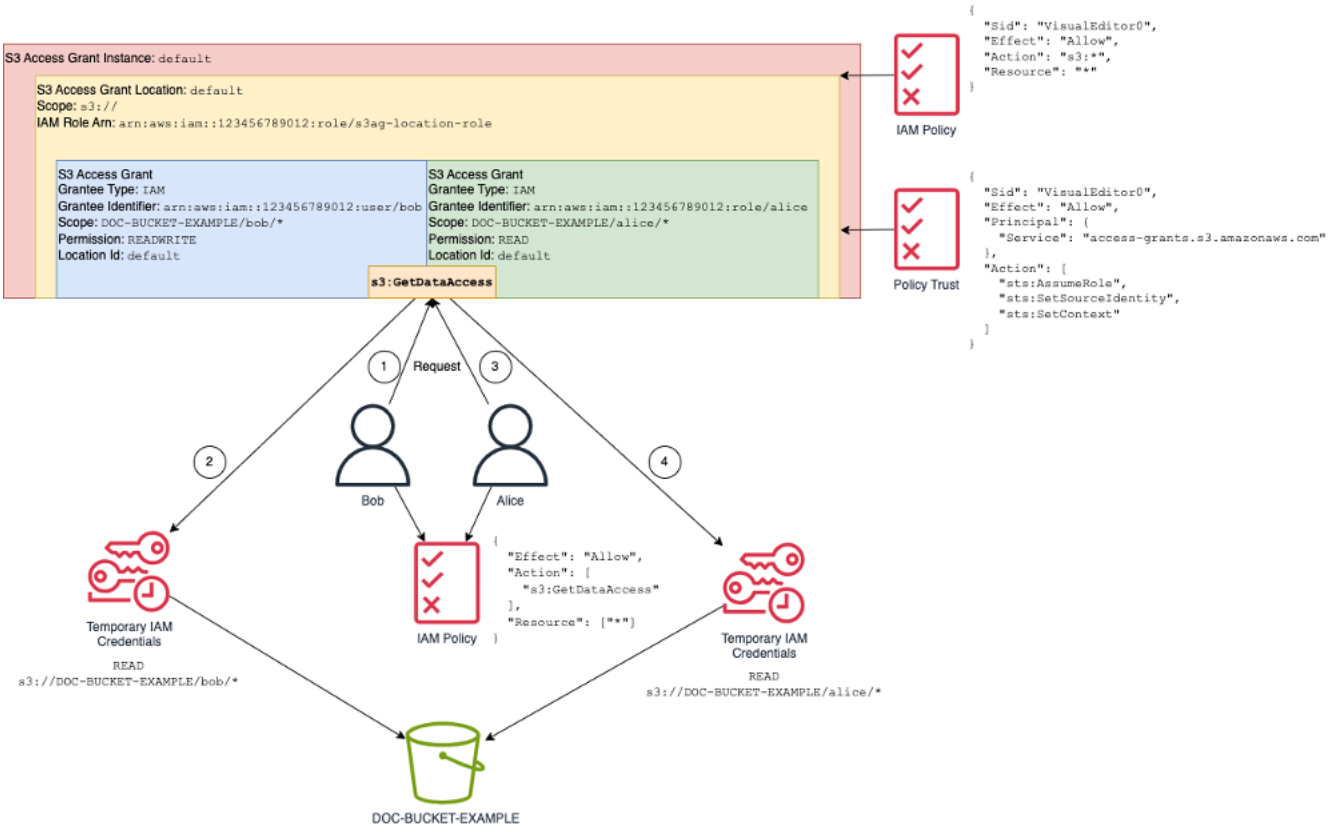
仕組み

次の図では、範囲が s3:// のデフォルトの Amazon S3 ロケーションが IAM ロール s3ag-location-role に登録されています。この IAM ロールには、S3 Access Grants を介して認証情報が取得されると、アカウント内で Amazon S3 アクションを実行するアクセス許可があります。

このロケーションでは、2 人の IAM ユーザーに対して 2 つの個別のアクセス権限が作成されます。IAM ユーザー Bob には、DOC-BUCKET-EXAMPLE バケット内のプレフィックス bob/ への READ と WRITE のアクセス権の両方が付与されます。別の IAM ロールである Alice には、DOC-BUCKET-EXAMPLE バケット内のプレフィックス alice/ への READ アクセス許可のみが付与されます。Bob が DOC-BUCKET-EXAMPLE バケット内のプレフィックス bob/ にアクセスするための権限は青色で定義されています。Alice が DOC-BUCKET-EXAMPLE バケット内のプレフィックス alice/ にアクセスするための権限は緑色で定義されています。

Bob がデータの READ を行う場合、Bob の権限が付与されているロケーションに関連付けられている IAM ロールが S3 Access Grants の GetDataAccess API オペレーションを呼び出します。Bob が s3://DOC-BUCKET-EXAMPLE/bob/* で始まる S3 プレフィックスまたはオブジェクトに対して READ を行おうとすると、GetDataAccess リクエストは s3://DOC-BUCKET-EXAMPLE/bob/* へのアクセス許可がある一時的な IAM セッション認証情報セットを返します。Bob は、権限により許可されているため、s3://DOC-BUCKET-EXAMPLE/bob/* で始まる任意の S3 プレフィックスまたはオブジェクトに対して WRITE を同様に実行できます。

同じように、Alice は、s3://DOC-BUCKET-EXAMPLE/alice/ で始まるすべてに対して READ を実行できます。ただし、s3:// 内のバケット、プレフィックス、オブジェクトに対して WRITE を実行しようすると、Access Denied (403 Forbidden) エラーが発生します。これは、どのデータに対する WRITE アクセスを Alice に付与する権限がないためです。さらに、Alice が s3://DOC-BUCKET-EXAMPLE/alice/ 外部のデータに対して何らかのアクセスレベル (READ または WRITE) をリクエストしても、同様に Access Denied エラーが表示されます。



このパターンは多数のユーザーとバケットに応じてスケールでき、アクセス許可の管理を簡素化します。個別のユーザーとプレフィックスのアクセス関係を追加または削除するたびに、サイズが大きくなる可能性のある S3 バケットポリシーを編集する代わりに、個別の権限を別々に追加したり削除したりできます。

S3 Access Grants と社内ディレクトリのアイデンティティ

Amazon S3 Access Grants を使用すると、同じ AWS アカウント 内と他のアカウントの両方で、AWS Identity and Access Management (IAM) プリンシパル (ユーザーまたはロール) へのアクセスを許可できます。ただし、多くの場合、データにアクセスするエンティティは社内ディレクトリのエンドユーザーです。IAM プリンシパルにアクセスを付与する代わりに、S3 Access Grants を使用して社内のユーザーやグループに直接アクセスを許可できます。S3 Access Grants を使用すると、社内アプリケーションを介して S3 データにアクセスするために、社内アイデンティティを中間となる IAM プリンシパルにマッピングする必要がなくなります。

エンドユーザーのアイデンティティの使用したデータへのアクセスをサポートするこの新機能は、S3 Access Grants インスタンスを AWS IAM Identity Center インスタンスに関連付けることで実現します。IAM アイデンティティセンターは、標準ベースのアイデンティティプロバイダーをサポートし、S3 Access Grants を含め、エンドユーザーアイデンティティをサポートするあらゆる

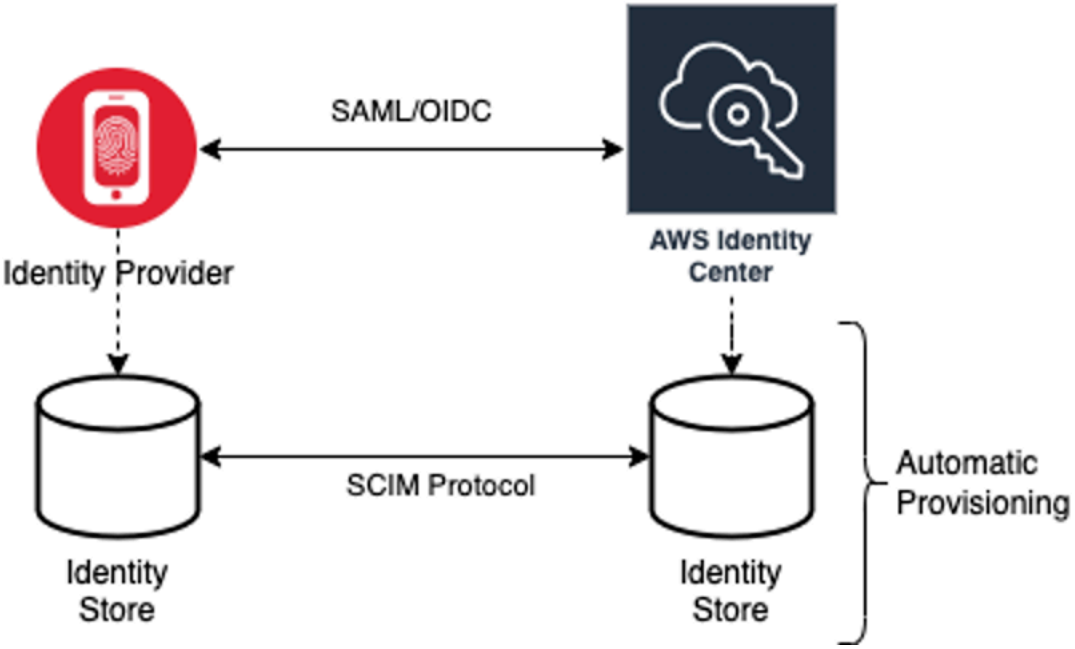
サービスや機能の AWS のハブとなります。IAM アイデンティティセンターは、信頼されているアイデンティティプロパゲーション機能を介して社内アイデンティティの認証をサポートします。詳細については、「アプリケーション間での信頼されたアイデンティティのプロパゲーション」を参照してください。

S3 Access Grants でワークフォースアイデンティティサポートの使用を開始するには、前提条件として、まず IAM アイデンティティセンターで、社内アイデンティティプロバイダーと IAM アイデンティティセンターの間のアイデンティティのプロビジョニングを設定します。IAM アイデンティティセンターは、Okta、Microsoft Entra ID (旧称 Azure Active Directory) などの社内アイデンティティプロバイダー、またはクロスドメインアイデンティティ管理システム (SCIM) プロトコルをサポートするその他の外部アイデンティティプロバイダー (IdP) をサポートしています。IAM アイデンティティセンターを IdP に接続して自動プロビジョニングを有効にすると、IdP のユーザーとグループが IAM アイデンティティセンターのアイデンティティストアに同期されます。このステップが完了すると、IAM アイデンティティセンターにはユーザーとグループに関する独自のビューが表示されるため、S3 Access Grants などのその他の AWS のサービス と機能を使用してユーザーやグループを参照できます。IAM アイデンティティセンターの自動プロビジョニングの設定に関する詳細は、「AWS IAM Identity Center ユーザーガイド」の「自動プロビジョニング」を参照してください。

IAM アイデンティティセンターは、AWS Organizations と統合されているため、各アカウントを手動で設定する必要なく、複数の AWS アカウント にわたるアクセス許可を一元管理できます。通常、組織では、アイデンティティ管理者が組織全体で単一の IAM アイデンティティセンターのインスタンスをアイデンティティ同期の単一ポイントとして、設定します。この IAM アイデンティティセンターのインスタンスは通常、組織内の専用 AWS アカウント インスタンスで実行されます。このような一般的な設定では、組織内のどの AWS アカウント からでも S3 Access Grants のユーザーアイデンティティとグループアイデンティティを参照できます。

ただし、AWS Organizations 管理者が中央の IAM アイデンティティセンターのインスタンスをまだ設定していない場合は、S3 Access Grants インスタンスと同じアカウントにローカルの IAM アイデンティティセンターインスタンスを作成できます。このような設定は、概念実証やローカルの開発環境でのユースケースで一般的です。いずれの場合も、IAM アイデンティティセンターのインスタンスは、関連付けられる S3 Access Grants インスタンスと同じ AWS リージョン に配置する必要があります。

次の図の外部 IdP を使用する IAM アイデンティティセンター設定では、IdP は SCIM で設定され、IdP のアイデンティティストアを IAM アイデンティティセンターのアイデンティティストアと同期します。



社内ディレクトリのアイデンティティを S3 Access Grants で使用するには、次を実行します。

- IAM アイデンティティセンターで 自動プロビジョニング を設定して、IdP からのユーザーとグループの情報を IAM アイデンティティセンターに同期します。
- IAM アイデンティティセンター内の外部アイデンティティソースを信頼できるトークン発行者として設定します。詳細については、「AWS IAM Identity Center ユーザーガイド」の「アプリケーション間での信頼されたアイデンティティのプロパゲーション」を参照してください。
- S3 Access Grants インスタンスを IAM アイデンティティセンターインスタンスに関連付けます。これは、S3 Access Grants インスタンス作成 時に実行できます。S3 Access Grants インスタンスをすでに作成している場合は、「IAM アイデンティティセンターインスタンスの関連付けまたは関連付けを解除します。」を参照してください。

ディレクトリのアイデンティティが S3 データにアクセスする方法

S3 データに社内アプリケーションを介してアクセスする必要がある社内ディレクトリのユーザーがいるとします。例えば、このアプリケーションは、(Okta などの) ユーザー認証を行う外部 IdP と統合されたドキュメントビューアだとします。これらのアプリケーションでのユーザーの認証は、通常、ユーザーのウェブブラウザのリダイレクトによって行われます。ディレクトリ内のユーザーは IAM プリンシパルではないため、アプリケーションには、S3 Access Grants `GetDataAccess` API オペレーションを呼び出し、ユーザーに代わって S3 データへのアクセス認証情報を取得 できる IAM 認証情報が必要です。認証情報を自身で取得する IAM ユーザーやロールとは異なり、アプリ

ケーションには IAM ロールにマップされていないディレクトリユーザーを示す方法が必要です。これにより、ユーザーは S3 Access Grants を通じてデータにアクセスできるようになります。

認証されたディレクトリユーザーから、ディレクトリユーザーに代わって S3 Access Grants へのリクエストを実行できる IAM 呼び出し元へのこの移行は、アプリケーションが IAM アイデンティティセンターの信頼できるトークン発行者機能を通じて行います。アプリケーションは、ディレクトリユーザーを認証すると、(Okta などの) IdP からのアイデンティティトークンを取得し、Okta に従ってディレクトリユーザーを示します。IAM アイデンティティセンターの信頼されたトークン発行者の設定により、アプリケーションはこの Okta トークン (Okta テナントが「信頼できる発行者」として設定されている) を、AWS のサービス 内でディレクトリユーザーを安全に示す IAM アイデンティティセンターの別のアイデンティティトークンと交換できます。その後、データアプリケーションは IAM ロールを引き受け、IAM アイデンティティセンターからのディレクトリユーザーのトークンを追加のコンテキストとして提供します。アプリケーションは、結果として得られる IAM セッションを使用して S3 Access Grants を呼び出すことができます。このトークンは、アプリケーションのアイデンティティ (IAM プリンシパル自体) とディレクトリユーザーのアイデンティティの両方を示します。

この移行の主なステップはトークンの交換です。アプリケーションは IAM アイデンティティセンターの `CreateTokenWithIAM` API オペレーションを呼び出して、このトークン交換を実行します。当然、これもまた AWS API 呼び出しであり、署名には IAM プリンシパルが必要です。このリクエストを行う IAM プリンシパルは、通常、アプリケーションに関連付けられた IAM ロールです。例えば、アプリケーションが Amazon EC2 で実行されている場合、`CreateTokenWithIAM` リクエストは通常、アプリケーションが実行されている EC2 インスタンスに関連付けられている IAM ロールによって実行されます。`CreateTokenWithIAM` 呼び出しが正常に完了すると、新しいアイデンティティトークンが生成され、AWS のサービス 内でそのトークンが認識されます。

アプリケーションがディレクトリユーザーに代わって `GetDataAccess` を呼び出す前の次のステップは、アプリケーションがディレクトリユーザーのアイデンティティを含む IAM セッションを取得することです。アプリケーションは、追加のアイデンティティコンテキストとしてディレクトリユーザーの IAM アイデンティティセンターのトークンも含めた AWS Security Token Service (AWS STS) `AssumeRole` リクエストを使用してこれを実行します。この追加コンテキストにより、IAM アイデンティティセンターはディレクトリユーザーのアイデンティティを次のステップにプロパゲートできるようになります。アプリケーションが引き受ける IAM ロールは、`GetDataAccess` オペレーションを呼び出すために IAM アクセス権限を必要とするロールです。

追加のコンテキストとしてディレクトリユーザーのための IAM アイデンティティセンターのトークンを使用して アイデンティティベアラー IAM ロールを引き受けたため、アプリケーションには、認

証されたディレクトリユーザーに代わって GetDataAccess に署名付きリクエストを行うために必要なものがすべて揃っています。

トークンのプロパゲーションは次のステップに基づいています。

IAM アイデンティティセンターのアプリケーションを作成する

まず、IAM アイデンティティセンターに新しいアプリケーションを作成します。このアプリケーションは、使用できるアプリケーション設定タイプを IAM アイデンティティセンターが識別できるテンプレートを使用します。アプリケーションを作成するコマンドでは、IAM アイデンティティセンターインスタンスの Amazon リソースネーム (ARN)、アプリケーション名、アプリケーションプロバイダー ARN を指定する必要があります。アプリケーションプロバイダーは、アプリケーションが IAM アイデンティティセンターを呼び出すために使用する SAML または OAuth アプリケーションプロバイダーです。

次のコマンド例を使用する際は、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws sso-admin create-application \
 --instance-arn "arn:aws:sso:::instance/ssoins-ssoins-1234567890abcdef" \
 --application-provider-arn "arn:aws:sso::aws:applicationProvider/custom" \
 --name MyDataApplication
```

レスポンス:

```
{
   "ApplicationArn": "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d"
}
```

信頼できるトークン発行者を作成します。

IAM アイデンティティセンターアプリケーションが完成したところで、次のステップは、IdP IdToken からの値を IAM アイデンティティセンターのトークンと交換するために使用される信頼できるトークン発行者を設定することです。このステップを完了するには、以下の項目が必要です。

- アイデンティティプロバイダーの発行者の URL
- 信頼されたトークン発行者名
- クレーム属性パス
- アイデンティティストア属性パス

- JSON Web Key Set (JWKS) 取り出しオプション

クレーム属性パスは、アイデンティティストア属性へのマッピングに使用されるアイデンティティプロバイダー属性です。通常、クレーム属性パスはユーザーのメールアドレスであるとはいえ、その他の属性を使用してマッピングを実行することもできます。

oidc-configuration.json というファイルを次の内容で作成します。このポリシーを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
  "OidcJwtConfiguration":
     {
      "IssuerUrl": "https://login.microsoftonline.com/a1b2c3d4-abcd-1234-b7d5-
b154440ac123/v2.0",
      "ClaimAttributePath": "preferred_username",
      "IdentityStoreAttributePath": "userName",
      "JwksRetrievalOption": "OPEN_ID_DISCOVERY"
     }
}
```

信頼できるトークン発行者を作成するには、次のコマンドを実行します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws sso-admin create-trusted-token-issuer \
  --instance-arn "arn:aws:sso:::instance/ssoins-1234567890abcdef" \
  --name MyEntraIDTrustedIssuer \
  --trusted-token-issuer-type OIDC_JWT \
  --trusted-token-issuer-configuration file://./oidc-configuration.json
```

レスポンス

```
{
  "TrustedTokenIssuerArn": "arn:aws:sso::123456789012:trustedTokenIssuer/
ssoins-1234567890abcdef/tti-43b4a822-1234-1234-1234-a1b2c3d41234"
}
```

IAM アイデンティティセンターアプリケーションを信頼できるトークン発行者に接続します。

信頼されたトークン発行者が機能するには、さらにいくつかの設定が必要です。信頼できるトークン発行者が信頼する対象者を設定します。対象者はキーによって識別される IdToken 内部の値で、アイデンティティプロバイダーの設定で確認できます。例:

```
1234973b-abcd-1234-abcd-345c5a9c1234
```

次のコンテンツを含む grant.json という名前のファイルを作成します。このファイルを使用するには、アイデンティティプロバイダーの設定と一致するように対象者を変更し、前のコマンドで返された信頼できるトークン発行者 ARN を指定します。

```
{
   "JwtBearer":
     {
       "AuthorizedTokenIssuers":
         [
           {
             "TrustedTokenIssuerArn": "arn:aws:sso::123456789012:trustedTokenIssuer/
ssoins-1234567890abcdef/tti-43b4a822-1234-1234-1234-a1b2c3d41234",
               "AuthorizedAudiences":
                 [
                   "1234973b-abcd-1234-abcd-345c5a9c1234"
                 ]
            }
         ]
     }
 }
```

次のコマンド例を実行します。このコマンドを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws sso-admin put-application-grant \
  --application-arn "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d" \
  --grant-type "urn:ietf:params:oauth:grant-type:jwt-bearer" \
  --grant file://./grant.json \
```

このコマンドは、grant.json ファイル内の対象者を信頼する設定を使用して、信頼されたトークン発行者を設定し、このオーディエンスを、タイプ jwt-bearer のトークンを交換するための最初のステップで作成されたアプリケーションにリンクします。urn:ietf:params:oauth:grant-type:jwt-bearer の文字列は任意の文字列ではありません。これは、OAuth SON Web Token (JWT) アサーションプロファイルに登録されている名前空間です。この名前空間の詳細については、RFC 7523 を参照してください。

次のコマンドを使用して、信頼できるトークン発行者が アイデンティティプロバイダーからの IdToken 値を交換する際にどの範囲を含めるかを設定します。S3 Access Grants の場合、--scope パラメータの値は s3:access_grants:read_write です。

```
aws sso-admin put-application-access-scope \
  --application-arn "arn:aws:sso::111122223333:application/ssoins-
ssoins-111122223333abcdef/apl-abcd1234a1b2c3d" \
  --scope "s3:access_grants:read_write"
```

最後のステップは、リソースポリシーを IAM アイデンティティセンターのアプリケーションにアタッチすることです。このポリシーにより、アプリケーションの IAM ロールが sso-oauth:CreateTokenWithIAM API オペレーションにリクエストを行い、IAM アイデンティティセンター から IdToken 値を受け取ることができるようになります。

次のコンテンツを含む authentication-method.json という名前のファイルを作成します。*123456789012* をアカウント ID に置き換えます。

```
{
   "Iam":
       {
         "ActorPolicy":
             {
                "Version": "2012-10-17",
                    "Statement":
                    [
                        {
                           "Effect": "Allow",
                            "Principal":
                            {
                              "AWS": "arn:aws:iam::123456789012:role/webapp"
                            },
                           "Action": "sso-oauth:CreateTokenWithIAM",
                            "Resource": "*"
                        }
                    ]
                }
            }
        }
```

ポリシーを IAM アイデンティティセンターアプリケーションにアタッチするには、次のコマンドを実行します。

```
aws sso-admin put-application-authentication-method \
   --application-arn "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d" \
   --authentication-method-type IAM \
   --authentication-method file://./authentication-method.json
```

これで、ウェブアプリケーションを通じてディレクトリユーザーに S3 Access Grants を使用するための設定が完了しました。この設定はアプリケーション内で直接テストすることも、IAM アイデンティティセンターアプリケーションポリシーで許可されている IAM ロールから次のコマンドを使用して CreateTokenWithIAM API オペレーションを呼び出しても行えます。

```
aws sso-oidc create-token-with-iam \
   --client-id "arn:aws:sso::123456789012:application/ssoins-ssoins-1234567890abcdef/
apl-abcd1234a1b2c3d"  \
   --grant-type urn:ietf:params:oauth:grant-type:jwt-bearer \
   --assertion IdToken
```

結果は次のとおりになります。

```
{
    "accessToken": "<suppressed long string to reduce space>",
    "tokenType": "Bearer",
    "expiresIn": 3600,
    "refreshToken": "<suppressed long string to reduce space>",
    "idToken": "<suppressed long string to reduce space>",
    "issuedTokenType": "urn:ietf:params:oauth:token-type:refresh_token",
    "scope": [
      "sts:identity_context",
      "s3:access_grants:read_write",
      "openid",
      "aws"
    ]
}
```

base64 でエンコードされた IdToken 値をデコードすると、JSON 形式のキーバリューペアが表示されます。キー sts:identity_context には、アプリケーションがディレクトリ ユーザーのアイデンティティ情報を含めるために sts:AssumeRole リクエストで送信する必要がある値が含まれています。デコードされた IdToken の例は次のとおりです。

```
{
    "aws:identity_store_id": "d-996773e796",
```

```
    "sts:identity_context": "AQoJb3JpZ2luX2VjEOTtl;<SUPRESSED>",
    "sub": "83d43802-00b1-7054-db02-f1d683aacba5",
    "aws:instance_account": "123456789012",
    "iss": "https://identitycenter.amazonaws.com/ssoins-1234567890abcdef",
    "sts:audit_context": "AQoJb3JpZ2luX2VjEOT<SUPRESSED>==",
    "aws:identity_store_arn": "arn:aws:identitystore::232642235904:identitystore/
d-996773e796",
    "aud": "abcd12344U0gi7n4Yyp0-WV1LWNlbnRyYWwtMQ",
    "aws:instance_arn": "arn:aws:sso:::instance/ssoins-6987d7fb04cf7a51",
    "aws:credential_id": "EXAMPLEHI5glPh40y9TpApJn8...",
    "act": {
       "sub": "arn:aws:sso::232642235904:trustedTokenIssuer/
ssoins-6987d7fb04cf7a51/43b4a822-1020-7053-3631-cb2d3e28d10e"
    },
    "auth_time": "2023-11-01T20:24:28Z",
    "exp": 1698873868,
    "iat": 1698870268
}
```

`sts:identity_context` から値を取得し、この情報を `sts:AssumeRole` 呼び出しで渡すことができます。CLI 出力例は次のとおりです。引き受けるロールは、`s3:GetDataAccess` 呼び出しアクセス許可のある一時的なロールです。

```
aws sts assume-role \
   --role-arn "arn:aws:iam::123456789012:role/temp-role" \
   --role-session-name "TempDirectoryUserRole" \
   --provided-contexts ProviderArn="arn:aws:iam::aws:contextProvider/
IdentityCenter",ContextAssertion="value from sts:identity_context"
```

これで、この呼び出しで受け取った認証情報を使用して `s3:GetDataAccess` API オペレーションを呼び出し、S3 リソースにアクセスできる最終的な認証情報を受け取ることができます。

S3 Access Grants の開始方法

Amazon S3 Access Grants は、S3 データにスケーラブルなアクセスコントロールソリューションを提供する Amazon S3 の機能です。S3 Access Grants は S3 認証情報ベンダーです。つまり、許可のリストとレベルを S3 Access Grants に登録することになります。その後、ユーザーまたはクライアントが S3 データにアクセスする必要が生じると、まず S3 Access Grants に認証情報を要求します。アクセスを許可する対応する権限がある場合、S3 Access Grants は一時的な最小特権のアクセス認証情報を送信します。その後、ユーザーまたはクライアントは S3 Access Grants から提供された認証情報を使用して S3 データにアクセスできます。これを踏まえて、S3 データ要件で複雑ま

たは大規模なアクセス許可設定が必要となる場合は、S3 Access Grants を使用して、ユーザー、グループ、ロール、アプリケーションの S3 データ権限をスケーリングできます。

ほとんどのユースケースでは、AWS Identity and Access Management (IAM) バケットポリシーまたは IAM アイデンティティベースのポリシーを使用して S3 データのアクセスコントロールを管理できます。

ただし、次のような複雑な S3 アクセスコントロール要件がある場合は、S3 Access Grants を使用すると多大な利点が得られます。

- バケットポリシーのサイズ制限である 20 KB に達している場合。
- 分析やビッグデータのために S3 データへのアクセス許可をヒューマンアイデンティティで付与している場合。例えば、Microsoft Entra ID (旧称 Azure Active Directory)、Okta、または Ping など。
- IAM ポリシーを頻繁に更新せずに、クロスアカウントアクセスを提供する必要がある場合。
- データが構造化されておらず、構造化されていない行と列の形式で、オブジェクトレベルの場合。

S3 Access Grants のワークフローは次のとおりです。

ステップ	説明
1	S3 Access Grants インスタンスを作成する 開始するには、個別のアクセス許可を含む S3 Access Grants インスタンスを起動します。
2	ロケーションを登録する 次に、S3 データロケーション (デフォルトの `s3://` など) を登録して、S3 データロケーションへのアクセスを提供する際に S3 Access Grants が引き受けるデフォルトの IAM ロールを指定します。また、特定のバケットまたはプレフィックスにカスタムロケーションを追加し、それらをカスタム IAM ロールにマッピングすることもできます。
3	権限を作成する 個別のアクセス許可権限を作成します。これらのアクセス権限では、登録されている S3 ロケーション、その ロケーション内

ステップ	説明
	のデータアクセス範囲、権限被付与者のアイデンティティ、アクセスレベル (READ、WRITE、またはREADWRITE) を指定します。
4	S3 データへのアクセスをリクエストする ユーザー、アプリケーション、S3 AWS のサービス データへのアクセスには、まずアクセスリクエストを行います。S3 Access Grants が、リクエストを承認すべきかどうかを決定します。アクセスを許可する対応する権限がある場合、S3 Access Grants は、その権限に関連付けられている登録済み口ケーションの IAM ロールを使用して、一時的な認証情報をリクエスタに返送します。
5	S3 データにアクセスする アプリケーションは S3 Access Grants が提供する一時的な認証情報を使用して S3 データにアクセスします。

S3 Access Grants インスタンスを作成する

Amazon S3 Access Grants の使用を開始するには、まず S3 Access Grants インスタンスを作成します。作成できるのは、アカウントにつき AWS リージョン ごとに単一の S3 Access Grants インスタンス のみです。S3 Access Grants インスタンスは、登録された口ケーションや権限を含む S3 Access Grants リソースのコンテナとして機能します。

S3 Access Grants を使用すると、AWS Identity and Access Management (IAM) ユーザーとロールに S3 データへのアクセス許可を付与することができます。AWS IAM Identity Center に 社内アイデンティティディレクトリを追加した場合は、社内ディレクトリのこの IAM アイデンティティセンターインスタンスを S3 Access Grants インスタンスに関連付けることができます。その後、社内ユーザーとグループにアクセス権限を作成できます。社内ディレクトリを IAM アイデンティティセンターにまだ追加していない場合は、後で S3 Access Grants インスタンスを IAM アイデンティティセンターインスタンスに関連付けることができます。

Amazon S3 コンソール、AWS Command Line Interface、(AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して S3 Storage Lens グループを作成および管理できます。

S3 コンソールの使用

S3 Access Grants を使用して S3 データへのアクセスを許可する前に、まず S3 データと同じ AWS リージョン に S3 Access Grants インスタンスを作成する必要があります。

前提条件

社内ディレクトリのアイデンティティを使用して S3 データへのアクセスを許可する場合は、AWS IAM Identity Center に社内アイデンティティディレクトリを追加します。まだ追加の準備が整っていない場合は、後で S3 Access Grants インスタンスを IAM アイデンティティセンターインスタンスに関連付けることができます。

S3 Access Grants インスタンスを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションバーで、現在表示されている AWS リージョン の名前をクリックします。次に、切り替え先のリージョンを選択します。
3. 左側のナビゲーションペインで、[Access Grants] を選択します。
4. [S3 Access Grants] ページで、[Create S3 Access Grants instance] をクリックします。

    a. [Access Grants インスタンスをセットアップ] ウィザードの [ステップ 1] で、現在の AWS リージョン でインスタンスを作成することを確認します。S3 データが配置されているのと同じ AWS リージョン であることを確認します。作成できるのは、アカウントにつき AWS リージョン ごとに単一の S3 Access Grants インスタンス です。

    b. (オプション) AWS IAM Identity Center に 社内アイデンティティディレクトリを追加した場合は、社内ディレクトリのこの IAM アイデンティティセンターインスタンスを S3 Access Grants インスタンスに関連付けることができます。

    これを実行するには、[Add IAM Identity Center instance in ***region***] を選択します。次に、IAM アイデンティティセンターインスタンスの Amazon リソースネーム (ARN) を入力します。

    社内ディレクトリを IAM アイデンティティセンターにまだ追加していない場合は、後で S3 Access Grants インスタンスを IAM アイデンティティセンターインスタンスに関連付けることができます。

    c. S3 Access Grants インスタンスを作成するには、[次へ] をクリックします。ロケーションの登録については、「ステップ 2 - ロケーションを登録する」を参照してください。

5. [次へ] または [Create S3 Access Grants instance] が無効になっている場合:

   インスタンスは作成できません

   - 同じ AWS リージョン に S3 Access Grants インスタンスが既にある可能性があります。左側のナビゲーションペインで、[Access Grants] を選択します。[S3 Access Grants] ページで、[S3 Access Grants instance in your account] セクションまで下にスクロールして、インスタンスが既に存在しているかを確認します。
   - S3 Access Grants インスタンスを作成するのに必要な `s3:CreateAccessGrantsInstance` アクセス許可がない可能性があります。アカウント管理者に連絡してください。IAM アイデンティティセンターインスタンスを S3 Access Grants インスタンスに関連付ける場合に必要な追加のアクセス許可については、「CreateAccessGrantsInstance」を参照してください。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

次のコマンド例を使用するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

Example S3 Access Grants インスタンスを作成する

```
aws s3control create-access-grants-instance \
--account-id 111122223333 \
--region us-east-2
```

レスポンス:

```
{
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00",
    "AccessGrantsInstanceId": "default",
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default"
}
```

REST API の使用

Amazon S3 REST API を使用して S3 Access Grants インスタンスを作成できます。REST API での ACL の管理のサポートの詳細については、「Amazon Simple Storage Service API リファレンス」の次のセクションを参照してください。

- AssociateAccessGrantsIdentityCenter
- CreateAccessGrantsInstance
- DeleteAccessGrantsInstance
- DissociateAccessGrantsIdentityCenter
- GetAccessGrantsInstance
- GetAccessGrantsInstanceForPrefix
- GetAccessGrantsInstanceResourcePolicy
- ListAccessGrantsInstances
- PutAccessGrantsInstanceResourcePolicy

AWS SDK の使用

このセクションでは、AWS SDK を使用して S3 Access Grants インスタンスを作成する方法の例を説明します。

Java

この例では、個別のアクセス許可のコンテナとして機能する S3 Access Grants インスタンスを作成します。作成できるのは、アカウントにつき AWS リージョン ごとに単一の S3 Access Grants インスタンスです。応答には、S3 Access Grants インスタンス用に生成されたインスタンス ID `default` と Amazon リソースネーム (ARN) が含まれます。

Example S3 Access Grants インスタンスリクエストを作成する

```
public void createAccessGrantsInstance() {
CreateAccessGrantsInstanceRequest createRequest =
 CreateAccessGrantsInstanceRequest.builder().accountId("111122223333").build();
CreateAccessGrantsInstanceResponse createResponse =
 s3Control.createAccessGrantsInstance(createRequest);LOGGER.info("CreateAccessGrantsInstance
 " + createResponse);
}
```

レスポンス:

```
CreateAccessGrantsInstanceResponse(
CreatedAt=2023-06-07T01:46:20.507Z,
AccessGrantsInstanceId=default,
AccessGrantsInstanceArn=arn:aws:s3:us-east-2:111122223333:access-grants/default)
```

トピック

- S3 Access Grants インスタンスの詳細を表示する
- IAM アイデンティティセンターインスタンスの関連付けまたは関連付けを解除します。
- S3 Access Grants インスタンスを削除する

S3 Access Grants インスタンスの詳細を表示する

特定の AWS リージョン の Amazon S3 Access Grants インスタンスの詳細を表示できます。AWS Resource Access Manager (AWS RAM) で共有されたインスタンスなど、S3 Access Grants インスタンスを一覧表示することもできます。

Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、Amazon S3 REST API、AWS SDK を使用して、S3 Access Grants インスタンスの詳細を表示したり、S3 Access Grants インスタンスを一覧表示できます。

S3 コンソールの使用

S3 Access Grants インスタンスを表示するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[Access Grants] を選択します。
3. [S3 Access Grants] ページで、取り組む S3 Access Grants インスタンスが含まれるリージョンを選択します。
4. [S3 Access Grants] ページには、S3 Access Grants インスタンスと、アカウントと共有されているクロスアカウントインスタンスが一覧表示されます。インスタンスの詳細を表示するには、[詳細の表示] を選択します。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

次のコマンド例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Example — S3 Access Grants インスタンスの詳細を取得する

```
aws s3control get-access-grants-instance \
 --account-id 111122223333 \
 --region us-east-2
```

レスポンス:

```
{
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default",
    "AccessGrantsInstanceId": "default",
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00"
}
```

Example — アカウントのすべての S3 Access Grants インスタンスを一覧表示する

このアクションでは、アカウントのすべての S3 Access Grants インスタンスを一覧表示します。AWS リージョン ごとに単一の S3 Access Grants インスタンスを持つことができます。このアクションでは、アカウントがアクセスできる他のクロスアカウント S3 Access Grants インスタンスも一覧表示されます。

```
aws s3control list-access-grants-instances \
 --account-id 111122223333 \
 --region us-east-2
```

レスポンス:

```
{
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default",
    "AccessGrantsInstanceId": "default",
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00"
```

```
}
```

REST API の使用

Amazon REST API での S3 Access Grants インスタンスの管理のサポートの詳細については、「Amazon Simple Storage Service API リファレンス」の次のセクションを参照してください。

- GetAccessGrantsInstance
- GetAccessGrantsInstanceForPrefix
- ListAccessGrantsInstances

AWS SDK の使用

このセクションでは、AWS SDK を使用して S3 Access Grants インスタンスの詳細を取得する方法の例を説明します。

次の例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Java

Example – S3 Access Grants インスタンスを取得する

```
public void getAccessGrantsInstance() {
GetAccessGrantsInstanceRequest getRequest = GetAccessGrantsInstanceRequest.builder()
.accountId("111122223333")
.build();
GetAccessGrantsInstanceResponse getResponse =
 s3Control.getAccessGrantsInstance(getRequest);
LOGGER.info("GetAccessGrantsInstanceResponse: " + getResponse);
}
```

レスポンス:

```
GetAccessGrantsInstanceResponse(
AccessGrantsInstanceArn=arn:aws:s3:us-east-2: 111122223333:access-grants/default,
CreatedAt=2023-06-07T01:46:20.507Z)
```

Example — アカウントのすべての S3 Access Grants インスタンスを一覧表示する

このアクションでは、アカウントのすべての S3 Access Grants インスタンスを一覧表示します。リージョンごとに単一の S3 Access Grants インスタンスを持つことができます。このアク

ションでは、アカウントがアクセスできる他のクロスアカウント S3 Access Grants インスタンスも一覧表示されます。

```
public void listAccessGrantsInstances() {
ListAccessGrantsInstancesRequest listRequest =
 ListAccessGrantsInstancesRequest.builder()
.accountId("111122223333")
.build();
ListAccessGrantsInstancesResponse listResponse =
 s3Control.listAccessGrantsInstances(listRequest);
LOGGER.info("ListAccessGrantsInstancesResponse: " + listResponse);
}
```

レスポンス:

```
ListAccessGrantsInstancesResponse(
AccessGrantsInstancesList=[
ListAccessGrantsInstanceEntry(
AccessGrantsInstanceId=default,
AccessGrantsInstanceArn=arn:aws:s3:us-east-2:111122223333:access-grants/default,
CreatedAt=2023-06-07T04:28:11.728Z
)
]
)
```

IAM アイデンティティセンターインスタンスの関連付けまたは関連付けを解除します。

Amazon S3 Access Grants では、企業アイデンティティディレクトリの AWS IAM Identity Center インスタンスを S3 Access Grants インスタンスに関連付けることができます。その後、社内ユーザーとグループ、AWS Identity and Access Management (IAM) ユーザーとロールにアクセス権限を作成できます。

社内ディレクトリのユーザーとグループにアクセス権限を作成する必要がなくなった場合は、IAM アイデンティティセンターインスタンスと S3 Access Grants インスタンスの関連付けを解除できます。

Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、Amazon S3 REST API、AWS SDK を使用して、IAM アイデンティティセンターインスタンスを関連付けたり、関連付けを解除したりできます。

S3 コンソールの使用

IAM アイデンティティセンターインスタンスを S3 Access Grants インスタンスに関連付ける前に、社内アイデンティティディレクトリを IAM アイデンティティセンターに追加する必要があります。詳細については、「the section called “S3 Access Grants と社内ディレクトリのアイデンティティ”」を参照してください。

IAM アイデンティティセンターインスタンスを S3 アクセス許可インスタンスに関連付けるには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[Access Grants] を選択します。
3. [S3 Access Grants] ページで、取り組む S3 Access Grants インスタンスが含まれるリージョンを選択します。
4. インスタンスの [詳細の表示] をクリックします。
5. 詳細ページの [IAM アイデンティティセンター] セクションで、IAM アイデンティティセンターインスタンスを [追加] するか、既に IAM アイデンティティセンターインスタンスに関連付けられているインスタンスを [登録解除] します。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

次のコマンド例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Example – IAM アイデンティティセンターインスタンスを S3 アクセス許可インスタンスに関連付ける

```
aws s3control associate-access-grants-identity-center \
 --account-id 111122223333 \
 --identity-center-arn arn:aws:sso:::instance/ssoins-1234a567bb89012c \
 --profile access-grants-profile \
 --region eu-central-1

// No response body
```

Example – IAM アイデンティティセンターインスタンスの S3 アクセス許可インスタンスへの関連付けを解除する

```
aws s3control dissociate-access-grants-identity-center \
 --account-id 111122223333 \
 --profile access-grants-profile \
 --region eu-central-1

// No response body
```

REST API の使用

Amazon S3 REST API での IAM アイデンティティセンターと S3 Access Grants 間の関連付けの管理のサポートの詳細については、「Amazon Simple Storage Service API リファレンス」の次のセクションを参照してください。

- AssociateAccessGrantsIdentityCenter
- DissociateAccessGrantsIdentityCenter

S3 Access Grants インスタンスを削除する

削除できるのは、アカウントの AWS リージョン の S3 Access Grants インスタンスです。ただし、S3 Access Grants インスタンスを削除する前に、まず次を完了する必要があります。

- S3 Access Grants インスタンス内の (すべての権限とロケーションを含む) すべてのリソースを削除します。詳細については、「権限を削除する」と「ロケーションを削除する」を参照してください。
- AWS IAM Identity Center インスタンスを S3 Access Grants インスタンスに関連付けた場合は、IAM アイデンティティセンターインスタンスの関連付けを解除する必要があります。詳細については、「IAM アイデンティティセンターインスタンスの関連付けまたは関連付けの解除」を参照してください。

> **Important**
>
> S3 Access Grants インスタンスの削除は永続的であり、元に戻すことができません。この S3 Access Grants インスタンスの権限を通じてアクセス権を付与されたすべての権限被付与者は、S3 データにアクセスできなくなります。

Amazon S3 コンソール、AWS Command Line Interface、(AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して S3 Storage Lens グループを削除できます。

S3 コンソールの使用

S3 Access Grants インスタンスを削除するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[Access Grants] を選択します。
3. [S3 Access Grants] ページで、取り組む S3 Access Grants インスタンスが含まれるリージョンを選択します。
4. インスタンスの [詳細の表示] をクリックします。
5. インスタンスの詳細ページで、右上隅の [インスタンスを削除] を選択します。
6. 表示されたダイアログボックスで、[削除] を選択します。この操作は元に戻すことができません。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

次のコマンド例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

> **Note**
>
> S3 Access Grants インスタンスを削除する前に、まず S3 Access Grants インスタンス内で作成されたすべての権限と口ケーションを削除する必要があります。IAM アイデンティティセンターインスタンスを S3 Access Grants インスタンスに関連付けている場合は、まずその関連付けを解除する必要があります。

Example – S3 Access Grants インスタンスを削除する

```
aws s3control delete-access-grants-instance \
--account-id 111122223333 \
```

```
--profile access-grants-profile \
--region us-east-2 \
--endpoint-url https://s3-control.us-east-2.amazonaws.com \

 // No response body
```

REST API の使用

Amazon REST API での S3 Access Grants インスタンスの削除のサポートの詳細については、「Amazon Simple Storage Service API リファレンス」の「DeleteAccessGrantsInstance」を参照してください。

AWS SDK の使用

このセクションでは、AWS SDK を使用して S3 Access Grants インスタンスを作成する方法の例を説明します。

次の例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Java

> **Note**
>
> S3 Access Grants インスタンスを削除する前に、まず S3 Access Grants インスタンス内で作成されたすべての権限と口ケーションを削除する必要があります。IAM アイデンティティセンターインスタンスを S3 Access Grants インスタンスに関連付けている場合は、まずその関連付けを解除する必要があります。

Example – S3 Access Grants インスタンスを削除する

```
public void deleteAccessGrantsInstance() {
DeleteAccessGrantsInstanceRequest deleteRequest =
 DeleteAccessGrantsInstanceRequest.builder()
.accountId("111122223333")
.build();
DeleteAccessGrantsInstanceResponse deleteResponse =
 s3Control.deleteAccessGrantsInstance(deleteRequest);
LOGGER.info("DeleteAccessGrantsInstanceResponse: " + deleteResponse);
}
```

ロケーションを登録する

アカウントの AWS リージョン で Amazon S3 Access Grants インスタンスを作成 したら、そのインスタンスに S3 ロケーションを登録できます。ロケーションは、アクセスを許可するデータを含む S3 リソースです。デフォルトの場所 s3:// (AWS リージョン 内のすべてのバケット) を登録し、後で個別のアクセス許可を作成する際にアクセスの範囲を限定することができます。特定のバケット、またはバケットとプレフィックスをロケーションとして登録することもできます。

アクセス権限を作成する前に、まず S3 Access Grants インスタンスに少なくとも 1 つのロケーションを登録する必要があります。ロケーションを登録する際は、S3 Access Grants がそのロケーションに対するランタイムリクエストを処理するために引き受ける AWS Identity and Access Management (IAM) ロールも指定する必要があります。また、実行時に特定のグラントに限定して権限の範囲を絞り込む必要があります。

S3 URI	IAM ロール	説明
s3://	*Default-IAM-role*	デフォルトのロケーション s3:// には、AWS リージョン 内のすべてのバケットが含まれます。
s3://*example-s3-bucket1* /	*IAM-role-For-bucket*	このロケーションには、指定したバケット内のすべてのオブジェクトが含まれます。

ロケーションを登録する前に、次の手順を実行する必要があります。

- アクセスを許可するデータを含むバケットを 1 つまたは複数作成します。これらのバケットは、S3 Access Grants インスタンスと同じ AWS リージョン にある必要があります。詳細については、「バケットの作成」を参照してください。

  バケットにプレフィックスを追加するには、「オブジェクトキー名の作成」を参照してください。
- IAM ロールを作成し、S3 Access Grants サービスプリンシパルにリソースポリシーファイル内のこのロールへのアクセス権を付与します。これを実行するために、次のステートメントを含む JSON ファイルを作成できます。リソースポリシーをアカウントに追加するには、「最初のカスタマー管理ポリシーの作成とアタッチ」を参照してください。

  TestRolePolicy.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1234567891011",
      "Action": ["sts:AssumeRole", "sts:SetSourceIdentity", "sts:SetContext"],
      "Effect": "Allow",
      "Principal": {"Service":"access-grants.s3.amazonaws.com"}
    }
  ]
}
```

- Amazon S3 バケットへのアクセス許可を持つ IAM ポリシーを作成します。次の `iam-policy.json` ファイルを例に s h じて、*`user input placeholders`* を実際の情報と置き換えます。

> **Note**
>
> - 次の例では、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化を使用してデータを暗号化する場合に、ポリシーに IAM ロールに必要な AWS KMS アクセス許可を含めています。この機能を使用しない場合は、IAM ポリシーからこのようなアクセス許可を削除できます。
> - IAM ロールが S3 データにアクセスするように制限できるのは、S3 Access Grants が認証情報を提供する場合に限られます。この例では、特定の S3 Access Grants インスタンスに `Condition` ステートメントを追加する方法を説明します。これを行うには、条件ステートメント内の S3 Access Grants インスタンス ARN を、`arn:aws:s3:`*`region`*`:`*`accountId`*`:access-grants/default` の形式の S3 Access Grants インスタンス ARN に置き換えます。

iam-policy.json

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
         "Sid": "ObjectLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
```

```
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetObjectAcl",
            "s3:GetObjectVersionAcl",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:region:accountId:access-
grants/default"]
            }
        }
      },
      {
         "Sid": "ObjectLevelWritePermissions",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionAcl",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:AbortMultipartUpload"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ####
#:accountId:access-grants/default"]
            }
         }
      },
      {
         "Sid": "BucketLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket"
```

```
            ],
            "Resource":[
               "arn:aws:s3:::*"
            ],
            "Condition":{
               "StringEquals": { "aws:ResourceAccount": "accountId" },
               "ArnEquals": {
                   "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ####
#:accountId:access-grants/default"]
               }
            }
         },
         {
            "Sid": "KMSPermissions",
            "Effect":"Allow",
            "Action":[
               "kms:Decrypt",
               "kms:GenerateDataKey"
            ],
            "Resource":[
               "*"
            ]
         }
      ]
}
```

Amazon S3 コンソール、AWS Command Line Interface、(AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して S3 Access Grants インスタンスに口ケーションを登録できます。

S3 コンソールの使用

S3 Access Grants を使用して S3 データへのアクセスを許可するには、少なくとも 1 つの口ケーションを登録しておく必要があります。

S3 Access Grants インスタンスで口ケーションを登録するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[Access Grants] を選択します。
3. [S3 Access Grants] ページで、取り組む S3 Access Grants インスタンスが含まれるリージョンを選択します。

S3 Access Grants インスタンスを初めて使用する場合は、「Step 1 - S3 Access Grants インスタンスを作成する」を実行し、[Access Grants インスタンスをセットアップ] ウィザードの [ステップ 2] に進んでいる必要があります。S3 Access Grants インスタンスが既にある場合は、[詳細の表示] を選択して、[ロケーション] タブで、[ロケーションを登録] をクリックします。

a. [ロケーション範囲] では、[Browse S3] をクリックするか、登録先のロケーションへの S3 URI パスを入力します。S3 URI の形式については、ロケーション形式 表を参照してください。URI の入力後、[表示] をクリックして、ロケーションを参照します。

b. [IAM ロール] で、次のいずれかを選択します。

- 既存の IAM ロールから選択

  ドロップダウンリストから IAM ロールを選択します。ロールを選択したら、[表示] を選択して、登録するロケーションを管理するのに必要なアクセス許可がこのロールにあることを確認します。具体的には、このロールが S3 Access Grants にアクセス許可 `sts:AssumeRole` と `sts:SetSourceIdentity` を付与していることを確認します。

- IAM ロールの ARN の入力

  [IAM コンソール] に入ります。IAM ロールの Amazon リソースネーム (ARN) をコピーして、このボックスに貼り付けます。

c. 終了するには、[次へ] または [ロケーションを登録] を選択します。

4. トラブルシューティング：

ロケーションを登録できない

- そのロケーションが既に登録されている可能性があります。

  ロケーションを登録する `s3:CreateAccessGrantsLocation` アクセス許可がない可能性があります。アカウント管理者に連絡してください。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

S3 Access Grants インスタンスには`s3://`、デフォルトの場所、またはカスタムロケーションを登録できます。その場所へのプリンシパルアクセス権を持つ IAM ロールを作成し、次に S3 Access Grants にこのロールを引き受ける権限を付与する必要があります。

次のコマンド例を使用する際は、*user input placeholders* をユーザー自身の情報に置き換えます。

Example リソースポリシーを作成する

S3 Access Grants が IAM ロールを引き受けることを許可するポリシーを作成します。これを実行するために、次のステートメントを含む JSON ファイルを作成できます。リソースポリシーをアカウントに追加するには、「最初のカスタマー管理ポリシーの作成とアタッチ」を参照してください。

TestRolePolicy.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1234567891011",
      "Action": ["sts:AssumeRole", "sts:SetSourceIdentity"],
      "Effect": "Allow",
      "Principal": {"Service":"access-grants.s3.amazonaws.com"}
    }
  ]
}
```

Example ロールを作成する

次の IAM コマンドを実行して、ロールを作成します。

```
aws iam create-role --role-name accessGrantsTestRole \
 --region us-east-2 \
 --assume-role-policy-document file://TestRolePolicy.json
```

`create-role` コマンドを実行すると、次のとおりポリシーが返されます。

```
{
    "Role": {
        "Path": "/",
        "RoleName": "accessGrantsTestRole",
        "RoleId": "AROASRDGX4WM4GH55GIDA",
        "Arn": "arn:aws:iam::111122223333:role/accessGrantsTestRole",
        "CreateDate": "2023-05-31T18:11:06+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
```

```
            "Statement": [
                {
                    "Sid": "Stmt1685556427189",
                    "Action": [
                        "sts:AssumeRole",
                        "sts:SetSourceIdentity"
                    ],
                    "Effect": "Allow",
                    "Principal": {
                        "Service":"access-grants.s3.amazonaws.com"
                    }
                }
            ]
        }
    }
}
```

Example

Amazon S3 バケットへのアクセス許可を持つ IAM ポリシーを作成します。次の `iam-policy.json` ファイルを例に s h じて、*`user input placeholders`* を実際の情報と置き換えます。

> **Note**
>
> AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化を使用してデータを暗号化する場合、次の例ではポリシーに IAM ロールに必要な AWS KMS アクセス許可を追加します。この機能を使用しない場合は、IAM ポリシーからこのようなアクセス許可を削除できます。
> 認証情報が S3 Access Grants によって提供される場合、IAM ロールが S3 内のデータへのアクセスにのみ使用できるようにするために、この例では、S3 Access Grants インスタンス (`s3:AccessGrantsInstance:` *`InstanceArn`*) を指定する `Condition` ステートメントを IAM ポリシーに追加する方法を示します。次のコマンド例を使用する際は、*`user input placeholders`* をユーザー自身の情報に置き換えます。

iam-policy.json

```
{
   "Version":"2012-10-17",
```

```
   "Statement": [
       {
         "Sid": "ObjectLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetObjectAcl",
            "s3:GetObjectVersionAcl",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:region:accountId:access-
grants/default"]
            }
        }
      },
      {
         "Sid": "ObjectLevelWritePermissions",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionAcl",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:AbortMultipartUpload"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS #####:accountId:access-
grants/default"]
            }
         }
      },
```

```
      {
         "Sid": "BucketLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS #####:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "KMSPermissions",
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt",
            "kms:GenerateDataKey"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Example

次のコマンドを実行します。

```
aws iam put-role-policy \
--role-name accessGrantsTestRole \
--policy-name accessGrantsTestRole \
--policy-document file://iam-policy.json
```

Example デフォルトロケーションを登録します。

```
aws s3control create-access-grants-location \
```

```
 --account-id 111122223333 \
 --location-scope s3:// \
 --iam-role-arn arn:aws:iam::111122223333:role/accessGrantsTestRole
```

レスポンス:

```
{"CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "LocationScope": "s3://”
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

Example カスタムロケーションを登録する

```
aws s3control create-access-grants-location \
 --account-id 111122223333 \
 --location-scope s3://DOC-BUCKET-EXAMPLE/ \
 --iam-role-arn arn:aws:iam::123456789012:role/accessGrantsTestRole
```

レスポンス:

```
{"CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb456",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://DOC-BUCKET-EXAMPLE/",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

REST API の使用

Amazon S3 REST API での S3 Access Grants インスタンスの管理のサポートの詳細については、「Amazon Simple Storage Service API リファレンス」の次のセクションを参照してください。

- CreateAccessGrantsLocation
- DeleteAccessGrantsLocation
- GetAccessGrantsLocation
- ListAccessGrantsLocations

- UpdateAccessGrantsLocation

AWS SDK の使用

このセクションでは、AWS SDK を使用して口ケーションを登録する方法の例を説明します。

次の例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Java

S3 Access Grants インスタンスにはs3://、デフォルトの場所、またはカスタム口ケーションを登録できます。その場所へのプリンシパルアクセス権を持つ IAM ロールを作成し、次に S3 Access Grants にこのロールを引き受ける権限を付与する必要があります。

次のコマンド例を使用する際は、*user input placeholders* をユーザー自身の情報に置き換えます。

Example デフォルトロケーションを登録する

リクエスト:

```
public void createAccessGrantsLocation() {
CreateAccessGrantsLocationRequest createRequest =
 CreateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.locationScope("s3://")
.iamRoleArn("arn:aws:iam::123456789012:role/accessGrantsTestRole")
.build();
CreateAccessGrantsLocationResponse createResponse =
 s3Control.createAccessGrantsLocation(createRequest);
LOGGER.info("CreateAccessGrantsLocationResponse: " + createResponse);
}
```

レスポンス:

```
CreateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:11.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope=s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
```

```
)
```

Example カスタムロケーションを登録する

リクエスト:

```
public void createAccessGrantsLocation() {
CreateAccessGrantsLocationRequest createRequest =
 CreateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.locationScope("s3://DOC-BUCKET-EXAMPLE/")
.iamRoleArn("arn:aws:iam::111122223333:role/accessGrantsTestRole")
.build();
CreateAccessGrantsLocationResponse createResponse =
 s3Control.createAccessGrantsLocation(createRequest);
LOGGER.info("CreateAccessGrantsLocationResponse: " + createResponse);
}
```

レスポンス:

```
CreateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=18cfe6fb-eb5a-4ac5-aba9-8d79f04c2012,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/18cfe6fb-eb5a-4ac5-aba9-8d79f04c2666,
LocationScope= s3://test-bucket-access-grants-user123/,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

トピック

- 登録済みロケーションの詳細を表示する
- 登録されたロケーションを更新する
- 登録されたロケーションを削除する

登録済みロケーションの詳細を表示する

Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、Amazon S3 REST API、AWS SDK を使用して、S3 Access Grants インスタンスに登録されているロケーションの詳細を取得できます。

S3 コンソールの使用

S3 Access Grants インスタンスに登録済みの口ケーションを表示するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[Access Grants] を選択します。
3. [S3 Access Grants] ページで、取り組む S3 Access Grants インスタンスが含まれるリージョンを選択します。
4. インスタンスの [詳細の表示] をクリックします。
5. インスタンスの詳細ページで、[口ケーション] タブをクリックします。
6. 確認する登録済みの口ケーションを検索します。登録済み口ケーションの一覧にフィルターを適用するには、検索ボックスを使用します。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

次のコマンド例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Example – 登録済み口ケーションの詳細を取得する

```
aws s3control get-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id default
```

レスポンス:

```
{
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

Example – S3 Access Grants インスタンスに登録済みの口ケーションをすべて一覧表示する

結果を S3 プレフィックスまたはバケットで限定するには、必要に応じて `--location-scope s3://bucket-and-or-prefix` パラメータを使用できます。

```
aws s3control list-access-grants-locations \
--account-id 111122223333 \
--region us-east-2
```

レスポンス:

```
{"AccessGrantsLocationsList": [
  {
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "LocationScope": "s3://"
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
     },
  {
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb456",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://DOC-EXAMPLE-BUCKET/prefixA*",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
     }
   ]
  }
```

REST API の使用

Amazon S3 REST API で登録済みの口ケーションの詳細を入手したり、S3 Access Grants インスタンスに登録された口ケーションのすべてを一覧表示する方法の詳細については、「Amazon Simple Storage Service API リファレンス」の次のセクションを参照してください。

- GetAccessGrantsLocation
- ListAccessGrantsLocations

AWS SDK の使用

このセクションでは、AWS SDK を使用して S3 Access Grants インスタンスに登録済みの口ケーションの詳細を取得したり、すべての登録済みの口ケーションを一覧表示する方法の例を説明します。

次の例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Java

Example – 登録済み口ケーションの詳細を取得する

```
public void getAccessGrantsLocation() {
GetAccessGrantsLocationRequest getAccessGrantsLocationRequest =
 GetAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("default")
.build();
GetAccessGrantsLocationResponse getAccessGrantsLocationResponse =
 s3Control.getAccessGrantsLocation(getAccessGrantsLocationRequest);
LOGGER.info("GetAccessGrantsLocationResponse: " + getAccessGrantsLocationResponse);
}
```

レスポンス:

```
GetAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope= s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

Example – S3 Access Grants インスタンスに登録済みの口ケーションをすべて一覧表示する

結果を S3 プレフィックスまたはバケットで限定するには、必要に応じて `LocationScope` パラメータで `s3://bucket-and-or-prefix` などの S3 URI を使用できます。

```
public void listAccessGrantsLocations() {
```

```
ListAccessGrantsLocationsRequest listRequest =
 ListAccessGrantsLocationsRequest.builder()
.accountId("111122223333")
.build();

ListAccessGrantsLocationsResponse listResponse =
 s3Control.listAccessGrantsLocations(listRequest);
LOGGER.info("ListAccessGrantsLocationsResponse: " + listResponse);
}
```

レスポンス:

```
ListAccessGrantsLocationsResponse(
AccessGrantsLocationsList=[
ListAccessGrantsLocationsEntry(
CreatedAt=2023-06-07T04:35:11.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope=s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
),
ListAccessGrantsLocationsEntry(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=635f1139-1af2-4e43-8131-a4de006eb456,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/635f1139-1af2-4e43-8131-a4de006eb888,
LocationScope=s3://DOC-EXAMPLE-BUCKET/prefixA*,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
]
)
```

登録されたロケーションを更新する

Amazon S3 Access Grants インスタンスに登録済みのロケーションの AWS Identity and Access Management (IAM) ロールは更新できます。S3 Access Grants にロケーションを登録するために使用する新しい IAM ロールごとに、このロールへの S3 Access Grants サービスプリンシパル (`access-grants.s3.amazonaws.com`) にアクセス権限を付与する必要があります。これを行うには、最初にロケーションを登録 した際に使用したのと同じ信頼ポリシー JSON ファイルに、新しい IAM ロールのエントリを追加します。

Amazon S3 コンソール、AWS Command Line Interface、(AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して S3 Access Grants インスタンスのロケーションを更新できます。

S3 コンソールの使用

Amazon S3 Access Grants インスタンスに登録済みのロケーションの IAM ロールを更新するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[Access Grants] を選択します。
3. [S3 Access Grants] ページで、取り組む S3 Access Grants インスタンスが含まれるリージョンを選択します。
4. インスタンスの [詳細の表示] をクリックします。
5. インスタンスの詳細ページで、[ロケーション] タブをクリックします。
6. 更新するロケーションを検索します。ロケーション一覧にフィルターを適用するには、検索ボックスを使用します。
7. 更新する登録済みのロケーションの隣にあるオプションボタンをクリックします。
8. IAM ロールを更新してから、[変更の保存] をクリックします。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

次のコマンド例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Example – 登録済みロケーションの IAM ロールを更新する

```
aws s3control update-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id 635f1139-1af2-4e43-8131-a4de006eb999 \
--iam-role-arn arn:aws:iam::777788889999:role/accessGrantsTestRole
```

レスポンス:

```
{
```

```
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb999",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:777788889999:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://DOC-EXAMPLE-BUCKET/prefixB*",
    "IAMRoleArn": "arn:aws:iam::777788889999:role/accessGrantsTestRole"
}
```

REST API の使用

Amazon S3 REST API での S3 Access Grants の ロケーション更新のサポートの詳細については、「Amazon Simple Storage Service API リファレンス」の「UpdateAccessGrantsLocation」を参照してください。

AWS SDK の使用

このセクションでは、AWS SDK を使用して登録済みの ロケーションを更新する方法の例を説明します。

次の例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Java

Example – 登録済み ロケーションの IAM ロールを更新する

```
public void updateAccessGrantsLocation() {
UpdateAccessGrantsLocationRequest updateRequest =
 UpdateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("635f1139-1af2-4e43-8131-a4de006eb999")
.iamRoleArn("arn:aws:iam::777788889999:role/accessGrantsTestRole")
.build();
UpdateAccessGrantsLocationResponse updateResponse =
 s3Control.updateAccessGrantsLocation(updateRequest);
LOGGER.info("UpdateAccessGrantsLocationResponse: " + updateResponse);
}
```

レスポンス:

```
UpdateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
```

```
AccessGrantsLocationId=635f1139-1af2-4e43-8131-a4de006eb999,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:777788889999:access-grants/default/
location/635f1139-1af2-4e43-8131-a4de006eb888,
LocationScope=s3://DOC-EXAMPLE-BUCKET/prefixB*,
IAMRoleArn=arn:aws:iam::777788889999:role/accessGrantsTestRole
)
```

登録されたロケーションを削除する

S3 Access Grants インスタンスからロケーション登録を削除することもできます。ロケーションを削除すると、S3 アクセス許可インスタンスから登録が解除されます。

S3 Access Grants インスタンスからロケーション登録を削除する前に、そのロケーションに関連付けられているすべての権限を削除する必要があります。権限を削除する方法については、「権限を削除する」を参照してください。

Amazon S3 コンソール、AWS Command Line Interface、(AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して S3 Access Grants インスタンスのロケーションを削除できます。

S3 コンソールの使用

S3 Access Grants インスタンスからロケーション登録を削除するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[Access Grants] を選択します。
3. [S3 Access Grants] ページで、取り組む S3 Access Grants インスタンスが含まれるリージョンを選択します。
4. インスタンスの [詳細の表示] をクリックします。
5. インスタンスの詳細ページで、[ロケーション] タブをクリックします。
6. 更新するロケーションを検索します。ロケーション一覧にフィルターを適用するには、検索ボックスを使用します。
7. 削除する登録済みのロケーションの隣にあるオプションボタンをクリックします。
8. [Deregister] (登録解除) を選択します。
9. この操作は元に戻せないことを警告するダイアログボックスが表示されます。ロケーションを削除するには、[登録解除] を選択します。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

次のコマンド例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Example — ロケーション登録を削除する

```
aws s3control delete-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id  a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
 // No response body
```

REST API の使用

Amazon S3 REST API での S3 Access Grants からのロケーション削除のサポートの詳細については、「Amazon Simple Storage Service API リファレンス」の「DeleteAccessGrantsLocation」を参照してください。

AWS SDK の使用

このセクションでは、AWS SDK を使用してロケーションを削除する方法の例を説明します。

次の例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Java

Example — ロケーション登録を削除する

```
public void deleteAccessGrantsLocation() {
DeleteAccessGrantsLocationRequest deleteRequest =
 DeleteAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("a1b2c3d4-5678-90ab-cdef-EXAMPLE11111")
.build();
DeleteAccessGrantsLocationResponse deleteResponse =
 s3Control.deleteAccessGrantsLocation(deleteRequest);
LOGGER.info("DeleteAccessGrantsLocationResponse: " + deleteResponse);
}
```

レスポンス:

```
DeleteAccessGrantsLocationResponse()
```

権限を作成する

Amazon S3 Access Grants で 少なくとも 1 つの口ケーションを登録 した後に、アクセス権限を作成できます。アクセス権限は、登録された口ケーションにアクセスするアクセス許可を被付与者に付与します。

被付与者は、AWS Identity and Access Management (IAM) ユーザーまたはロール、またはディレクトリユーザーまたはグループにすることができます。ディレクトリ ユーザーは、S3 Access Grants インスタンスに関連付けられている AWS IAM Identity Center インスタンスに追加した、社内ディレクトリまたは外部アイデンティティソースのユーザーです。IAM アイデンティティセンターから特定のユーザーまたはグループに対する権限を作成するには、IAM アイデンティティセンターでそのユーザーを識別するために使用する GUID (例: a1b2c3d4-5678-90ab-cdef-EXAMPLE11111) を検索します。

バケット、プレフィックス、またはオブジェクトへのアクセスを付与できます。Amazon S3 のプレフィックスは、バケット内のオブジェクトを整理するために使用されるオブジェクトキー名の先頭にある文字列です。これには、使用できる任意の文字列を使用できます。例えば、engineering/プレフィックスで始まるバケット内のオブジェクトキー名などです。

サブプレフィックス

登録した口ケーションへのアクセスを付与する場合、Subprefix フィールドを使用して範囲をバケット内の特定のプレフィックスまたはバケット内の特定のオブジェクトに絞り込むことができます。

デフォルト口ケーション s3:// に対しては、被付与者がリージョンのすべてのバケットにアクセスできるアクセス権限を作成することはできません。デフォルト口ケーション s3:// の場所を付与場所として選択した場合は、Subprefix フィールドを使用して次のいずれかを指定して付与範囲を狭める必要があります。

- バケット — s3://*bucket*/*
- バケット内のプレフィックス — s3://*bucket*/*prefix**
- プレフィックス内のプレフィックス — s3://*bucket*/*prefixA*/*prefixB**
- オブジェクト — s3://*bucket*/*object-key-name*

登録済みロケーションがバケットであるアクセス権限を作成する場合、Subprefix フィールドには次のいずれかを渡すことができます。

- バケット内のプレフィックス — *prefix**
- プレフィックス内のプレフィックス — *prefixA/prefixB**
- オブジェクト — */object-key-name*

Amazon S3 コンソールに表示される権限範囲、または API や AWS Command Line Interface (AWS CLI) レスポンスで返される GrantScope は、ロケーションパスと Subprefix を連結した結果です。この連結されたパスが、アクセスを許可する S3 バケット、プレフィックス、またはオブジェクトに適切にマップされていることを確認します。

単一のオブジェクトのみへのアクセスを許可するアクセス権限を作成する場合は、API コールまたは CLI コマンドで s3PrefixType を Object と指定します。

> **Note**
>
> バケットがまだ存在していない場合、そのバケットへの許可は作成できません。ただし、まだ存在しないプレフィックスに対して許可を作成することはできます。

Amazon S3 コンソール、AWS CLI、Amazon S3 REST API、AWS SDK を使用して アクセス権限を作成できます。

S3 コンソールの使用

アクセス権限を作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[Access Grants] を選択します。
3. [S3 Access Grants] ページで、取り組む S3 Access Grants インスタンスが含まれるリージョンを選択します。

   S3 Access Grants インスタンスを初めて使用する場合は、「Step 2 - ロケーションを登録する」を実行し、Access Grants インスタンスをセットアップ ウィザードの [ステップ 3] に進んでいる必要があります。S3 Access Grants インスタンスが既にある場合は、[詳細の表示] を選択して、[権限] タブで、[権限を作成] をクリックします。

a. [権限範囲] セクションで、登録済みの口ケーションを選択または入力します。

デフォルトロケーションの s3:// を選択した場合は、[サブプレフィックス] ボックスを使用してアクセス権限の範囲を絞り込むことができます。詳細については、「サブプレフィックス」を参照してください。単一のオブジェクトのみへのアクセスを付与するには、[Grant scope is an object] を選択します。

b. [許可とアクセス] で、[アクセス許可] レベルに [読み取り]、[書き込み]、または両方を選択します。

次に、[被付与者タイプを選択] をクリックします。社内ディレクトリを IAM アイデンティティセンターに追加して、この IAM アイデンティティセンターインスタンスを S3 Access Grants インスタンスに関連付けた場合は、[IAM アイデンティティセンターからのディレクトリ ID] を選択できます。このオプションを選択した場合は、IAM アイデンティティセンターからユーザーまたはグループのアイデンティティを取得し、このセクションに入力します。

[被付与者タイプ] が IAM ユーザーまたはロールの場合は、[IAM プリンシパル] を選択します。[IAM プリンシパルタイプ] で [ユーザー] または [ロール] を選択します 次に、[IAM プリンシパルユーザー] で、リストから選択するか、アイデンティティを入力します。

c. S3 Access Grants 権限を作成するには、[次へ] または [権限を作成] をクリックします。

4. [次へ] または [権限を作成] が無効になっている場合:

権限が作成できない

- まず、S3 Access Grants インスタンスで 口ケーションを登録 する必要がある場合があります。
- アクセス権限を作成するのに必要な s3:CreateAccessGrant アクセス許可がない可能性があります。アカウント管理者に連絡してください。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

IAM プリンシパルのアクセス許可リクエストを作成する方法と、社内ディレクトリのユーザーまたはグループにアクセス権限リクエストを作成する方法の例は、次のとおりです。

次のコマンド例を使用する際は、*user input placeholders* をユーザー自身の情報に置き換えます。

> **Note**
>
> 単一のオブジェクトのみへのアクセスを許可するアクセス権限を作成する場合は、必要な `--s3-prefix-type Object` パラメータを含めます。

Example IAM プリンシパルのアクセス権限リクエストを作成する

```
aws s3control create-access-grant \
--account-id 111122223333 \
--access-grants-location-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222 \
--access-grants-location-configuration S3SubPrefix=prefixB* \
--permission READ \
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::123456789012:user/data-consumer-3
```

Example アクセス権限のレスポンスを作成する

```
{"CreatedAt": "2023-05-31T18:41:34.663000+00:00",
    "AccessGrantId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Grantee": {
        "GranteeType": "IAM",
        "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
    },
    "AccessGrantsLocationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "AccessGrantsLocationConfiguration": {
        "S3SubPrefix": "prefixB*"
    },
    "GrantScope": "s3://DOC-BUCKET-EXAMPLE/prefix*",
    "Permission": "READ"
}
```

ディレクトリユーザーやグループのためのアクセス権限リクエストを作成する

ディレクトリユーザーまたはグループのアクセス許可リクエストを作成するには、まず次のコマンドのいずれかを実行してディレクトリユーザーまたはグループの GUID を取得する必要があります。

Example ディレクトリユーザーやグループの GUID を取得する

IAM アイデンティティセンターユーザーの GUID は、IAM アイデンティティセンターコンソール、AWS CLI、または AWS SDK を使用して検索できます。次のコマンドは、指定した IAM アイデンティティセンターインスタンス内のユーザーを名前とアイデンティティとともに一覧表示します。

```
aws identitystore list-users --identity-store-id d-1a2b3c4d1234
```

このコマンドは、指定した IAM アイデンティティセンターインスタンスのグループを一覧表示します。

```
aws identitystore list-groups --identity-store-id d-1a2b3c4d1234
```

Example ディレクトリユーザーまたはグループのためのアクセス権限を作成する

このコマンドは IAM ユーザーまたはロールの権限の作成と似ています。ただし、被付与者タイプは DIRECTORY_USER または DIRECTORY_GROUP で、被付与者識別子がディレクトリユーザーまたはグループの GUID である点が異なります。

```
aws s3control create-access-grant \
--account-id 123456789012 \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix="DOC-EXAMPLE-BUCKET/rafael/*" \
--permission READWRITE \
--grantee GranteeType=DIRECTORY_USER,GranteeIdentifier=83d43802-00b1-7054-db02-
f1d683aacba5 \
```

REST API の使用

REST API でのアクセス権限管理のサポートの詳細については、「Amazon Simple Storage Service API リファレンス」の次のセクションを参照してください。

- CreateAccessGrant
- DeleteAccessGrant
- GetAccessGrant
- ListAccessGrants

AWS SDK の使用

このセクションでは、AWS SDK を使用してアクセス権限を作成する方法の例を説明します。

Java

次のコマンド例を使用する際は、*user input placeholders* をユーザー自身の情報に置き換えます。

> **Note**
>
> 単一のオブジェクトのみへのアクセスを許可するアクセス権限を作成する場合は、必要な `.s3PrefixType(S3PrefixType.Object)` パラメータを含めます。

Example アクセス権限リクエストを作成する

```
public void createAccessGrant() {
CreateAccessGrantRequest createRequest = CreateAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa")
.permission("READ")
.accessGrantsLocationConfiguration(AccessGrantsLocationConfiguration.builder().s3SubPrefix("
.grantee(Grantee.builder().granteeType("IAM").granteeIdentifier("arn:aws:iam::111122223333:u
data-consumer-3").build())
.build();
CreateAccessGrantResponse createResponse =
 s3Control.createAccessGrant(createRequest);
LOGGER.info("CreateAccessGrantResponse: " + createResponse);
}
```

Example アクセス権限のレスポンスを作成する

```
CreateAccessGrantResponse(
CreatedAt=2023-06-07T05:20:26.330Z,
AccessGrantId=a1b2c3d4-5678-90ab-cdef-EXAMPLE33333,
AccessGrantArn=arn:aws:s3:us-east-2:444455556666:access-grants/default/grant/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333,
Grantee=Grantee(
GranteeType=IAM,
GranteeIdentifier=arn:aws:iam::111122223333:user/data-consumer-3
),
AccessGrantsLocationId=a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa,
AccessGrantsLocationConfiguration=AccessGrantsLocationConfiguration(
S3SubPrefix=prefixB*
```

```
),
GrantScope=s3://DOC-BUCKET-EXAMPLE/prefixB,
Permission=READ
)
```

トピック

- 権限を表示する
- 権限を削除する

権限を表示する

Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、Amazon S3 REST API、AWS SDK を使用して、S3 Access Grants インスタンスのアクセス権限の詳細を表示できます。

S3 コンソールの使用

アクセス権限の詳細を表示するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[Access Grants] を選択します。
3. [S3 Access Grants] ページで、取り組む S3 Access Grants インスタンスが含まれるリージョンを選択します。
4. インスタンスの [詳細の表示] をクリックします。
5. [詳細] ページで、[権限] タブをクリックします。
6. [権限] セクションで、確認するアクセス権限を検索します。権限一覧にフィルターを適用するには、検索ボックスを使用します。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

次のコマンド例を使用する際は、*user input placeholders* をユーザー自身の情報に置き換えます。

Example — アクセス権限の詳細を取得する

```
aws s3control get-access-grant \
--account-id 111122223333 \
--access-grant-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222
```

レスポンス:

```
{
    "CreatedAt": "2023-05-31T18:41:34.663000+00:00",
    "AccessGrantId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Grantee": {
        "GranteeType": "IAM",
        "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
    },
    "Permission": "READ",
    "AccessGrantsLocationId": "12a6710f-5af8-41f5-b035-0bc795bf1a2b",
    "AccessGrantsLocationConfiguration": {
        "S3SubPrefix": "prefixB*"
    },
    "GrantScope": "s3://DOC-EXAMPLE-BUCKET/"
}
```

Example – S3 Access Grants インスタンスのアクセス権限をすべて一覧表示する

必要に応じて次のパラメータを使用して、結果を S3 プレフィックスまたは AWS Identity and Access Management (IAM) ID に限定できます。

- サブプレフィックス – `--grant-scope s3://bucket-name/prefix*`
- IAM アイデンティティ – `--grantee-type IAM` と `--grantee-identifier arn:aws:iam::123456789000:role/accessGrantsConsumerRole`

```
aws s3control list-access-grants \
--account-id 111122223333
```

レスポンス:

```
{
```

```
    "AccessGrantsList": [{"CreatedAt": "2023-06-14T17:54:46.542000+00:00",
            "AccessGrantId": "dd8dd089-b224-4d82-95f6-975b4185bbaa",
            "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/dd8dd089-b224-4d82-95f6-975b4185bbaa",
            "Grantee": {
                "GranteeType": "IAM",
                "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
            },
            "Permission": "READ",
            "AccessGrantsLocationId": "23514a34-ea2e-4ddf-b425-d0d4bfcarda1",
            "GrantScope": "s3://DOC-EXAMPLE-BUCKET/prefixA*"
        },
        {"CreatedAt": "2023-06-24T17:54:46.542000+00:00",
            "AccessGrantId": "ee8ee089-b224-4d72-85f6-975b4185a1b2",
            "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/ee8ee089-b224-4d72-85f6-975b4185a1b2",
            "Grantee": {
                "GranteeType": "IAM",
                "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-9"
            },
            "Permission": "READ",
            "AccessGrantsLocationId": "12414a34-ea2e-4ddf-b425-d0d4bfcacao0",
            "GrantScope": "s3://DOC-EXAMPLE-BUCKET/prefixB*"
        },

    ]
}
```

REST API の使用

Amazon S3 API オペレーションを使用して、アクセス権限の詳細を表示し、S3 Access Grants インスタンス内のすべてのアクセス権限を一覧表示できます。REST API でのアクセス権限管理のサポートの詳細については、「Amazon Simple Storage Service API リファレンス」の次のセクションを参照してください。

- GetAccessGrant
- ListAccessGrants

AWS SDK の使用

このセクションでは、AWS SDK を使用してアクセス権限の詳細を取得する方法の例を説明します。

次の例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Java

Example — アクセス権限の詳細を取得する

```
public void getAccessGrant() {
GetAccessGrantRequest getRequest = GetAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantId("a1b2c3d4-5678-90ab-cdef-EXAMPLE22222")
.build();
GetAccessGrantResponse getResponse = s3Control.getAccessGrant(getRequest);
LOGGER.info("GetAccessGrantResponse: " + getResponse);
}
```

レスポンス:

```
GetAccessGrantResponse(
CreatedAt=2023-06-07T05:20:26.330Z,
AccessGrantId=a1b2c3d4-5678-90ab-cdef-EXAMPLE22222,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant-fd3a5086-42f7-4b34-9fad-472e2942c70e,
Grantee=Grantee(
GranteeType=IAM,
GranteeIdentifier=arn:aws:iam::111122223333:user/data-consumer-3
),
Permission=READ,
AccessGrantsLocationId=12a6710f-5af8-41f5-b035-0bc795bf1a2b,
AccessGrantsLocationConfiguration=AccessGrantsLocationConfiguration(
S3SubPrefix=prefixB*
),
GrantScope=s3://DOC-EXAMPLE-BUCKET/
)
```

Example – S3 Access Grants インスタンスのアクセス権限をすべて一覧表示する

必要に応じて次のパラメータを使用して、結果を S3 プレフィックスまたは IAM アイデンティティに限定できます。

- 範囲 – `GrantScope=s3://bucket-name/prefix*`

- 被付与者 – GranteeType=IAM と GranteeIdentifier= arn:aws:iam::*111122223333*:role/*accessGrantsConsumerRole*

```
public void listAccessGrants() {
ListAccessGrantsRequest listRequest = ListAccessGrantsRequest.builder()
.accountId("111122223333")
.build();
ListAccessGrantsResponse listResponse = s3Control.listAccessGrants(listRequest);
LOGGER.info("ListAccessGrantsResponse: " + listResponse);
}
```

レスポンス:

```
ListAccessGrantsResponse(
AccessGrantsList=[
ListAccessGrantEntry(
CreatedAt=2023-06-14T17:54:46.540z,
AccessGrantId=dd8dd089-b224-4d82-95f6-975b4185bbaa,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/dd8dd089-b224-4d82-95f6-975b4185bbaa,
Grantee=Grantee(
GranteeType=IAM, GranteeIdentifier= arn:aws:iam::111122223333:user/data-consumer-3
),
Permission=READ,
AccessGrantsLocationId=23514a34-ea2e-4ddf-b425-d0d4bfcarda1,
GrantScope=s3://DOC-EXAMPLE-BUCKET/prefixA
),
ListAccessGrantEntry(
CreatedAt=2023-06-24T17:54:46.540Z,
AccessGrantId=ee8ee089-b224-4d72-85f6-975b4185a1b2,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/ee8ee089-b224-4d72-85f6-975b4185a1b2,
Grantee=Grantee(
GranteeType=IAM, GranteeIdentifier= arn:aws:iam::111122223333:user/data-consumer-9
),
Permission=READ,
AccessGrantsLocationId=12414a34-ea2e-4ddf-b425-d0d4bfcacao0,
GrantScope=s3://DOC-EXAMPLE-BUCKET/prefixB*
)
]
)
```

権限を削除する

S3 Access Grants インスタンスからアクセス権限を削除することもできます。アクセス権限の削除は元に戻すことができません。アクセス権限を削除すると、被付与者は Amazon S3 データにアクセスできなくなります。

Amazon S3 コンソール、AWS Command Line Interface、(AWS CLI)、AWS SDK、または Amazon S3 REST API を使用してアクセス権限を削除できます。

S3 コンソールの使用

アクセス権限を削除するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[Access Grants] を選択します。
3. [S3 Access Grants] ページで、取り組む S3 Access Grants インスタンスが含まれるリージョンを選択します。
4. インスタンスの [詳細の表示] をクリックします。
5. [詳細] ページで、[権限] タブをクリックします。
6. 削除する権限を検索します。権限を見つけたら、その横にあるラジオボタンをオンにします。
7. [削除] を選択します。この操作は元に戻せないことを警告するダイアログボックスが表示されます。もう一度 [削除] をクリックすると、権限が削除されます。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

次のコマンド例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Example – アクセス権限を削除する

```
aws s3control delete-access-grant \
--account-id 111122223333 \
```

```
--access-grant-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111

// No response body
```

REST API の使用

Amazon REST API でのアクセス権限管理のサポートの詳細については、「Amazon Simple Storage Service API リファレンス」の「DeleteAccessGrant」を参照してください。

AWS SDK の使用

このセクションでは、AWS SDK を使用してアクセス権限を削除する方法の例を説明します。次の例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Java

Example – アクセス権限を削除する

```
public void deleteAccessGrant() {
DeleteAccessGrantRequest deleteRequest = DeleteAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantId("a1b2c3d4-5678-90ab-cdef-EXAMPLE11111")
.build();
DeleteAccessGrantResponse deleteResponse =
 s3Control.deleteAccessGrant(deleteRequest);
LOGGER.info("DeleteAccessGrantResponse: " + deleteResponse);
}
```

レスポンス:

```
DeleteAccessGrantResponse()
```

S3 Access Grants を介して Amazon S3 データへのアクセスをリクエストする

Amazon S3 Access Grants を使用して、AWS Identity and Access Management (IAM) プリンシパル、社内ディレクトリアイデンティティ、または認証済みアプリケーションに S3 データへのアクセスを付与するアクセス権限を作成 した後、被付与者はこのデータにアクセススルタンの認証情報をリクエストできます。

アプリケーションまたは AWS のサービス API `GetDataAccess` オペレーションを使用して、被付与者に代わって S3 Access Grants に S3 データへのアクセスを要求すると、S3 Access Grants はま

ず、ユーザーがこのアイデンティティにデータへのアクセスを付与しているかを確認します。次に S3 Access Grants は AssumeRole API オペレーションを使用して一時的な認証トークンを取得し、それをリクエスタに送信します。この一時的な認証情報トークンは AWS Security Token Service (AWS STS) トークンです。

GetDataAccess リクエストには、一時的な認証情報が適用される S3 target データの範囲を指定するパラメータを含める必要があります。この target 範囲は権限の範囲と同一でも、その範囲のサブセットでもかまいません。ただし、target 範囲はリクエスタに与えられた権限の範囲内である必要があります。このリクエストでは、一時認証情報の権限レベル (READ、WRITE、READWRITE、など) を示す permission パラメータも指定する必要があります。

リクエスタは、認証情報リクエストで一時トークンの特権レベルを指定できます。リクエスタはこの privilege パラメータを使用して、一時的な認証情報のアクセス範囲を付与範囲内で拡大または縮小できます。privilege パラメータのデフォルト値は Default であり、返される認証情報のターゲット範囲は元の権限範囲です。privilege でこれ以外に指定できる値は、Minimal です。target 範囲が元の権限範囲から縮小される場合、target 範囲が権限範囲内にある限り、一時的な認証情報は target 範囲と一致するように範囲が再定義されます。

2 つの権限に対する privilege パラメータの効果の詳細は、次のテーブルのとおりです。一方の権限の範囲は、S3://*example-s3-bucket1*/bob/*で、*example-s3-bucket1* バケットの bob/ プレフィックス全体が含まれます。もう一方の権限の範囲は、S3://*example-s3-bucket1*/bob/reports/* で、*example-s3-bucket1* バケットの bob/reports/ プレフィックスのみが含まれます。

権限範囲	リクエスト範囲	特権	返される範囲	効果
S3://*example-s3-bucket1* /bob/*	*example-s3-bucket1* /bob/*	Default	*example-s3-bucket1* /bob/*	リクエスタは、*example-s3-bucket1* バケット内のプレフィックス *bob/* で始まるキー名を持つすべてのオブジェクトにアクセスできます。
S3://*example-s3-*	*example-s3-*	Minimal	*example-s3-bucket1* /bob/	プレフィックス名 bob/ の後にワイルドカードの * 文字がないと、リク

権限範囲	リクエスト範囲	特権	返される範囲	効果
bucket1 /bob/*	*bucket1* /bob/			エスタがアクセスできるのは、*example-s3-bucket1* バケット内のオブジェクト名 bob/ のみです。このようなオブジェクトは通常存在しません。リクエスタは、bob/ プレフィックスで始まるキー名を持つオブジェクトを含め、その他のオブジェクトにはアクセスできません。
S3://*example-s3-bucket1* /bob/*	*example-s3-bucket1* /bob/images/*	Minimal	*example-s3-bucket1* /bob/images/*	リクエスタは、*example-s3-bucket1* バケット内のプレフィックス *bob/images/** で始まるキー名を持つすべてのオブジェクトにアクセスできます。
S3://*example-s3-bucket1* /bob/reports/*	*example-s3-bucket1* /bob/reports/file.txt	Default	*example-s3-bucket1* /bob/reports/*	リクエスタは、*example-s3-bucket1* バケット内の bob/reports プレフィックスで始まるキー名を持つすべてのオブジェクトにアクセスできます。これは、権限と一致する範囲です。

権限範囲	リクエスト範囲	特権	返される範囲	効果
S3://*example-s3-bucket1* /bob/reports/*	*example-s3-bucket1* /bob/reports/file.txt	Minimal	*example-s3-bucket1* /bob/reports/file.txt	リクエスタは、*example-s3-bucket1* バケット内のキー名 bob/reports/file.txt のオブジェクトにのみアクセスできます。リクエスタは、その他のオブジェクトにはアクセスできません。

durationSeconds パラメータでは一時認証情報の有効期間を秒単位で設定します。デフォルト値は 3600 秒 (1 時間) です。リクエスタ (被付与者) は 900 秒 (15 分) から 43200 秒 (12 時間) までの範囲を指定できます。被付与者がこの最大値よりも高い値をリクエストすると、そのリクエストは失敗します。

> **Note**
>
> 一時トークンのリクエストで、ロケーションがオブジェクトの場合は、targetType リクエスト内のパラメータの値を Object に設定します。このパラメータは、ロケーションがオブジェクトで、特権レベルが Minimal の場合にのみ必要です。ロケーションがバケットまたはプレフィックスの場合、このパラメータを指定する必要はありません。

詳細については、「Amazon Simple Storage Service ユーザーガイド」の「GetDataAccess」を参照してください。

AWS Command Line Interface (AWS CLI)、Amazon S3 REST API、AWS SDK を使用して一時的な認証情報をリクエストできます。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

次のコマンド例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Example 一時認証情報のリクエスト

リクエスト:

```
aws s3control get-data-access \
--account-id 111122223333 \
--target s3://example-s3-bucket/prefixA* \
--permission READ \
--privilege Default \
--region us-east-2
```

レスポンス:

```
{
"Credentials": {
 "AccessKeyId": "Example-key-id",
 "SecretAccessKey": "Example-access-key",
 "SessionToken": "Example-session-token",
"Expiration": "2023-06-14T18:56:45+00:00"},
    "MatchedGrantTarget": "s3://example-s3-bucket/prefixA**"
}
```

REST API の使用

Amazon S3 REST API での S3 Access Grants からの一時認証情報のリクエストのサポートの詳細については、「Amazon Simple Storage Service API リファレンス」の「https://docs.aws.amazon.com/AmazonS3/latest/API/API_control_GetDataAccess.html」を参照してください。

AWS SDK の使用

このセクションでは、AWS SDK を使用して被付与者が一時認証情報をリクエストする方法の例を説明します。

Java

次のコード例は、被付与者が S3 データにアクセスするために使用する一時的な認証情報を返します。次のコマンド例を使用する際は、*user input placeholders* をユーザー自身の情報に置き換えます。

Example 一時認証情報を入手する

リクエスト:

```
public void getDataAccess() {
GetDataAccessRequest getDataAccessRequest = GetDataAccessRequest.builder()
.accountId("111122223333")
.permission(Permission.READ)
.privilege(Privilege.MINIMAL)
.target("s3://example-s3-bucket/prefixA*")
.build();
GetDataAccessResponse getDataAccessResponse =
 s3Control.getDataAccess(getDataAccessRequest);
LOGGER.info("GetDataAccessResponse: " + getDataAccessResponse);
}
```

レスポンス:

```
GetDataAccessResponse(
Credentials=Credentials(
AccessKeyId="Example-access-key-id",
SecretAccessKey="Example-secret-access-key",
SessionToken="Example-session-token",
Expiration=2023-06-07T06:55:24Z
))
```

アクセス権限を介して S3 データにアクセスします。

被付与者がアクセス許可を通じて一時的な認証情報を取得 すると、その一時的な認証情報を使用して Amazon S3 API オペレーションを呼び出し、データにアクセスできます。

被付与者は、AWS Command Line Interface (AWS CLI)、AWS SDK、Amazon S3 REST API を使用して、S3 データにアクセスできます。

AWS CLI の使用

被付与者は S3 Access Grants から一時的な認証情報を取得したら、その認証情報を使用してプロファイルを設定してデータを取得できます。

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

次のコマンド例を使用する際は、*user input placeholders* をユーザー自身の情報に置き換えます。

Example – プロファイルを設定する

```
aws configure set aws_access_key_id "$accessKey" --profile access-grants-consumer-access-profile
aws configure set aws_secret_access_key "$secretKey" --profile access-grants-consumer-access-profile
aws configure set aws_session_token "$sessionToken" --profile access-grants-consumer-access-profile
```

次のコマンド例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Example – S3 データを取得する

被付与者は、get-object AWS CLI コマンドを使用して、データにアクセスできます。被付与者は、put-object、ls、その他の S3 AWS CLI コマンドも使用できます。

```
aws s3api get-object \
--bucket example-s3-bucket1 \
--key myprefix \
--region us-east-2 \
--profile access-grants-consumer-access-profile
```

AWS SDK の使用

このセクションでは、AWS SDK を使用して S3 データにアクセスする方法の例を説明します。

Java

一時的な認証情報を使用して S3 データを取得する方法の例については、「AWSSDK を使用してオブジェクトを取得する方法」と「AWS SDK for Java 2.x の Amazon S3 コード例」を参照してください。

S3 Access は、クロスアカウントアクセスを許可します。

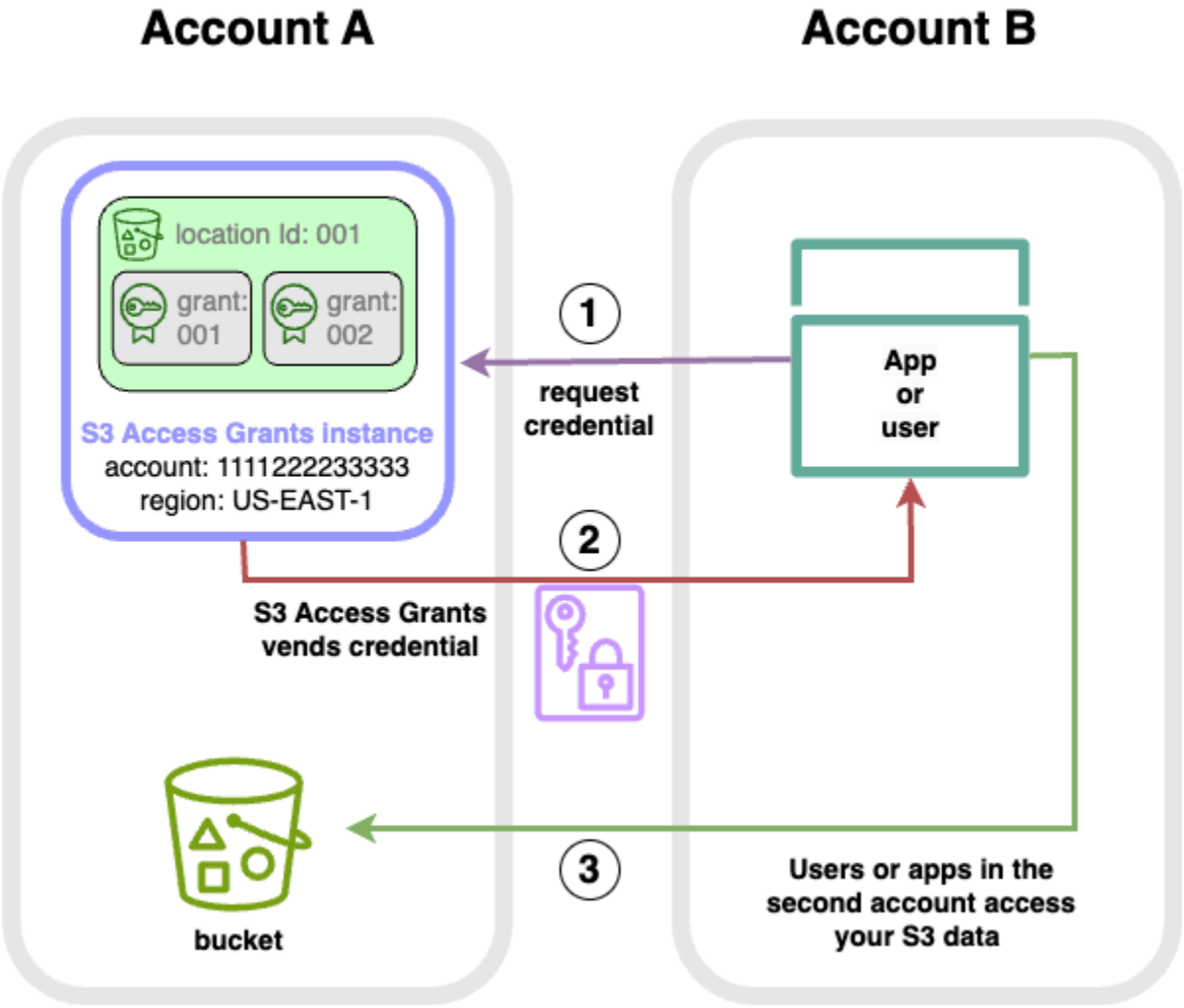
S3 Access Grants を使用すると、Amazon S3 データアクセスを以下に付与できます。

- AWS Identity and Access Management アカウント内の (IAM) アイデンティティ

- 他の AWS アカウントの IAM アイデンティティ
- AWS IAM Identity Center インスタンス内のディレクトリユーザーまたはグループ

まず、他のアカウントでのクロスアカウントアクセスを設定します。これには、リソースポリシーを使用して S3 Access Grants インスタンスへのアクセス許可の付与などがあります。次に、許可を使用して S3 データ (バケット、プレフィックス、オブジェクト) へのアクセス許可を付与します。

クロスアカウントアクセスの設定後、他のアカウントは S3 Access Grants から Amazon S3 データへの一時的なアクセス認証情報をリクエストできるようになります。次の図は、S3 Access Grants によるクロスアカウント S3 アクセスのユーザーフローを説明しています。



1. 2 番目のアカウント (B) のユーザーまたはアプリケーションは、Amazon S3 データが保存されているアカウント (A) の S3 Access Grants インスタンスに認証情報をリクエストします。詳細については、「S3 Access Grants を介して Amazon S3 データへのアクセスをリクエストする」を参照してください。

2. アカウント (A) の S3 Access Grants インスタンスは、2 番目のアカウントに Amazon S3 データへのアクセスを付与する許可がある場合、一時的な認証情報を返します。詳細については、「the section called “権限を作成する”」を参照してください。
3. 2 番目のアカウント (B) のユーザーまたはアプリケーションは、S3 Access Grants が提供する認証情報を使用して、アカウント (A) の S3 データにアクセスします。

S3 Access Grants クロスアカウントアクセスの設定

S3 Access Grants を介してクロスアカウント S3 アクセスを付与するには、次の手順を実行します。

- ステップ 1: S3 データが保存されているアカウント ID 111122223333 など、S3 Access Grants インスタンスをアカウントに設定します。
- ステップ 2: アカウント 111122223333 の S3 Access Grants インスタンスのリソースポリシーを設定して、2 番目のアカウント (アカウント ID 444455556666 など) へのアクセスを許可します。
- ステップ 3: 2 番目のアカウント 444455556666 の IAM プリンシパルの IAM アクセス許可を設定し、アカウント 111122223333 の S3 Access Grants インスタンスに認証情報をリクエストします。
- ステップ 4: 2 番目のアカウント 444455556666 の IAM プリンシパルに 111122223333 アカウント内の S3 データの一部へのアクセスを付与する許可を 111122223333 アカウントに作成します。

ステップ 1: アカウントで S3 Access Grants インスタンスを設定する

Amazon S3 へのアクセスを管理するには、まずアカウント 111122223333 に S3 Access Grants インスタンスが必要です。共有する S3 データが保存されている各 AWS リージョン に S3 Access Grants インスタンスを作成する必要があります。データを複数の AWS リージョン 間で共有する場合は、各 AWS リージョン について、この設定手順を繰り返します。共有する S3 データが保存されている AWS リージョン に S3 Access Grants インスタンスが既にある場合は、次のステップに進みます。S3 Access Grants インスタンスを設定していない場合は、「S3 Access Grants インスタンスを作成する」を参照してこのステップを完了します。

ステップ 2: クロスアカウントアクセスを許可するに S3 Access Grants インスタンスを許可するように、S3 Access Grants インスタンスのリソースポリシーを設定する

アカウント 111122223333 でクロスアカウントアクセス用の S3 Access Grants インスタンスを作成したら、アカウント 111122223333 内の S3 Access Grants インスタンスのリソースベースのポリシーを設定して、クロスアカウントアクセスを許可します。S3 Access Grants インスタンス自体は、リソースベースのポリシーをサポートしています。適切なリソースベースのポリシーを施行すると、その他の AWS アカウント の AWS Identity and Access Management (IAM) ユーザーまたはロールに S3 Access Grants インスタンスへのアクセスを付与できます。クロスアカウントアクセスで付与されるのは、次のアクセス許可 (アクション) のみです。

- `s3:GetAccessGrantsInstanceForPrefix` — ユーザー、ロール、またはアプリは、特定のプレフィックスを持つ S3 Access Grants インスタンスを取得できます。
- `s3:ListAccessGrants`
- `s3:ListAccessLocations`
- `s3:GetDataAccess` — ユーザー、ロール、またはアプリは、S3 Access Grants を介して付与されたアクセスに基づいて一時的な認証情報をリクエストできます。これらの認証情報を使用して、アクセスが付与されている S3 データにアクセスする。

このようなアクセス許可のいずれかをリソースポリシーに含めることができます。S3 Access Grants インスタンスのこのリソースポリシーは、通常のリソースベースのポリシーであり、IAM ポリシー言語がサポートするすべてをサポートします。同じポリシーで、例えば `aws:PrincipalArn` 条件を使用して、アカウント 111122223333 の特定の IAM アイデンティティにアクセスを付与することはできるとはいえ、S3 Access Grants ではその必要はありません。代わりに、S3 Access Grants インスタンス内で、自身のアカウントおよびその他のアカウントの個別の IAM アイデンティティに対する付与を作成できます。S3 Access Grants を使用して各アクセス権限を個別に管理することにより、アクセス許可をスケールできます。

AWS Resource Access Manager (AWS RAM) を既に使用している場合は、これを使用して、`s3:AccessGrants` リソースをその他のアカウントや組織内で共有できます。詳細については、「共有 AWS リソースの使用」を参照してください。AWS RAM を使用しない場合は、S3 Access Grants API オペレーションと AWS Command Line Interface (AWS CLI) を使用してリソースポリシーを追加することもできます。

S3 コンソールの使用

`s3:AccessGrants` リソースをその他のアカウントや組織内で共有するには、AWS Resource Access Manager (AWS RAM) コンソールを使用することをお勧めします。S3 Access Grants をクロスアカウントで共有するには、以下を実行します。

S3 Access Grants インスタンスのリソースポリシーを設定するには:

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. AWS リージョン セレクタから AWS リージョン を選択します。
3. 左側のナビゲーションペインで、[Access Grants] を選択します。
4. Access Grants インスタンスページの [このアカウントのインスタンス] セクションで、[インスタンスを共有] をクリックします。これにより、AWS RAM コンソールにリダイレクトされます。
5. [リソース共有を作成] をクリックします。
6. AWS RAM ステップを実行して、リソース共有を作成します。詳細については、「AWS RAM ユーザーガイド」の「リソース共有の作成」を参照してください。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

リソースポリシーは、`put-access-grants-instance-resource-policy` CLI コマンドを使用して追加できます。

アカウント `111122223333` の S3 Access Grants インスタンスへのクロスアカウントアクセスを 2 番目のアカウント `444455556666` に付与する場合、アカウント `111122223333` の S3 Access Grants インスタンスのリソースポリシーは、2 番目のアカウント `444455556666` に次のアクションを実行するアクセス許可を付与する必要があります。

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

S3 Access Grants インスタンスのリソースポリシーで、S3 Access Grants インスタンスの ARN を Resource として指定し、2 番目のアカウント 444455556666 を Principal として指定します。次の例を使用する際は、*###############*を独自の情報に置き換えます。

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

S3 Access Grants インスタンスのリソースポリシーを追加または更新するには、次のコマンドを使用します。次のコマンド例を使用する際は、*user input placeholders* を独自の情報に置き換えます。

Example S3 Access Grants インスタンスのリソースポリシーを追加または更新する

```
aws s3control put-access-grants-instance-resource-policy \
--account-id 111122223333 \
--policy file://resourcePolicy.json \
--region us-east-2
{
 "Policy": "{\n
   \"Version\": \"2012-10-17\",\n
   \"Statement\": [{\n
  \"Effect\": \"Allow\",\n
  \"Principal\": {\n
    \"AWS\": \"444455556666\"\n
  },\n
  \"Action\": [\n
```

```
    \"s3:ListAccessGrants\",\n
    \"s3:ListAccessGrantsLocations\",\n
    \"s3:GetDataAccess\",\n
    \"s3:GetAccessGrantsInstanceForPrefix\"\n
  ],\n
  \"Resource\": \"arn:aws:s3:us-east-2:111122223333:access-grants/default\"\n
    }\n
   ]\n
   }\n",
 "CreatedAt": "2023-06-16T00:07:47.473000+00:00"
}
```

Example S3 Access Grants リソースポリシーを取得する

CLI を使用して、S3 Access Grants インスタンスのリソースポリシーを取得または削除することもできます。

S3 Access Grants リソースポリシーを取得するには、次のサンプルコマンドを使用します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control get-access-grants-instance-resource-policy \
--account-id 111122223333 \
--region us-east-2

{
"Policy": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",\"Principal
\":{\"AWS\":\"arn:aws:iam::111122223333:root\"},\"Action\":[\"s3:ListAccessGrants\",
\"s3:ListAccessGrantsLocations\",\"s3:GetDataAccess\"],\"Resource\":\"arn:aws:s3:us-
east-2:111122223333:access-grants/default\"}]}",
"CreatedAt": "2023-06-16T00:07:47.473000+00:00"
}
```

Example S3 Access Grants リソースポリシーを削除する

S3 Access Grants リソースポリシーを削除するには、次のサンプルコマンドを使用します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control delete-access-grants-instance-resource-policy \
--account-id 111122223333 \
--region us-east-2
```

```
// No response body
```

REST API の使用

PutAccessGrantsInstanceResourcePolicy API を使用してリソースポリシーを追加できます。

アカウント 111122223333 の S3 Access Grants インスタンスへのクロスアカウントアクセスを 2 番目のアカウント 444455556666 に付与する場合、アカウント 111122223333 の S3 Access Grants インスタンスのリソースポリシーは、2 番目のアカウント 444455556666 に次のアクションを実行するアクセス許可を付与する必要があります。

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

S3 Access Grants インスタンスのリソースポリシーで、S3 Access Grants インスタンスの ARN を `Resource` として指定し、2 番目のアカウント 444455556666 を `Principal` として指定します。次の例を使用する際は、*###############*を独自の情報に置き換えます。

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

その後、PutAccessGrantsInstanceResourcePolicy API を使用して、ポリシーを設定できます。

S3 Access Grants インスタンスのリソースポリシーを更新、取得、または削除するための REST API サポートの詳細については、「Amazon Simple Storage Service API リファレンス」の次のセクションを参照してください。

- PutAccessGrantsInstanceResourcePolicy
- GetAccessGrantsInstanceResourcePolicy
- DeleteAccessGrantsInstanceResourcePolicy

AWS SDK の使用

このセクションでは、S3 アクセス許可リソース ポリシーを設定して 2 番目の AWS アカウントに S3 データの一部へのアクセスを許可する方法の AWS SDK の例を示します。

Java

リソースポリシーを追加、更新、取得、または削除して、S3 Access Grants インスタンスへのクロスアカウントアクセスを管理します。

Example S3 Access Grants インスタンスのリソースポリシーを追加または更新する

アカウント 111122223333 の S3 Access Grants インスタンスへのクロスアカウントアクセスを 2 番目のアカウント 444455556666 に付与する場合、アカウント 111122223333 の S3 Access Grants インスタンスのリソースポリシーは、2 番目のアカウント 444455556666 に次のアクションを実行するアクセス許可を付与する必要があります。

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

S3 Access Grants インスタンスのリソースポリシーで、S3 Access Grants インスタンスの ARN を `Resource` として指定し、2 番目のアカウント `444455556666` を `Principal` として指定します。次の例を使用する際は、`###############`を独自の情報に置き換えます。

```
{
"Version": "2012-10-17",
"Statement": [
{
```

```
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

S3 Access Grants インスタンスのリソースポリシーを追加または更新するには、次のサンプルコードを使用します。

```
public void putAccessGrantsInstanceResourcePolicy() {
 PutAccessGrantsInstanceResourcePolicyRequest putRequest =
 PutAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .policy(RESOURCE_POLICY)
 .build();
 PutAccessGrantsInstanceResourcePolicyResponse putResponse =
 s3Control.putAccessGrantsInstanceResourcePolicy(putRequest);
 LOGGER.info("PutAccessGrantsInstanceResourcePolicyResponse: " + putResponse);
 }
```

レスポンス:

```
PutAccessGrantsInstanceResourcePolicyResponse(
 Policy={
 "Version": "2012-10-17",
 "Statement": [{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
 },
 "Action": [
 "s3:ListAccessGrants",
 "s3:ListAccessGrantsLocations",
 "s3:GetDataAccess",
 "s3:GetAccessGrantsInstanceForPrefix"
```

```
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
 }]
 }
 )
```

Example S3 Access Grants リソースポリシーを取得する

S3 Access Grants リソースポリシーを取得するには、次のサンプルコマンドを使用します。次のコマンド例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
public void getAccessGrantsInstanceResourcePolicy() {
 GetAccessGrantsInstanceResourcePolicyRequest getRequest =
 GetAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .build();
 GetAccessGrantsInstanceResourcePolicyResponse getResponse =
 s3Control.getAccessGrantsInstanceResourcePolicy(getRequest);
 LOGGER.info("GetAccessGrantsInstanceResourcePolicyResponse: " + getResponse);
 }
```

レスポンス:

```
GetAccessGrantsInstanceResourcePolicyResponse(
 Policy={"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"AWS":"arn:aws:iam::444455556666:root"},"Action":
["s3:ListAccessGrants","s3:ListAccessGrantsLocations","s3:GetDataAccess"],"Resource":"arn:aw
east-2:111122223333:access-grants/default"}]},
 CreatedAt=2023-06-15T22:54:44.319Z
 )
```

Example S3 Access Grants リソースポリシーを削除する

S3 Access Grants リソースポリシーを削除するには、次のサンプルコマンドを使用します。次のコマンド例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
public void deleteAccessGrantsInstanceResourcePolicy() {
 DeleteAccessGrantsInstanceResourcePolicyRequest deleteRequest =
 DeleteAccessGrantsInstanceResourcePolicyRequest.builder()
```

```
.accountId(111122223333)
.build();
DeleteAccessGrantsInstanceResourcePolicyResponse deleteResponse =
s3Control.putAccessGrantsInstanceResourcePolicy(deleteRequest);
LOGGER.info("DeleteAccessGrantsInstanceResourcePolicyResponse: " + deleteResponse);
}
```

レスポンス:

```
DeleteAccessGrantsInstanceResourcePolicyResponse()
```

ステップ 3: 2 番目のアカウントの IAM アイデンティティに、アカウント内の S3 Access Grants インスタンスを呼び出すためのアクセス許可を付与する

Amazon S3 データの所有者がアカウント 111122223333 の S3 Access Grants インスタンスのクロスアカウントポリシーを設定した後、2 番目のアカウント 444455556666 の所有者は、IAM ユーザーまたはロールのアイデンティティベースのポリシーを作成し、S3 Access Grants インスタンスへのアクセスを付与する必要があります。S3 Access Grants インスタンスのリソースポリシーで付与されている内容と付与するアクセス許可に応じて、アイデンティティベースのポリシーに次のアクションを 1 つまたは複数含めます。

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

「AWS cross-account access pattern」に従って、2 番目のアカウント 444455556666 の IAM ユーザーまたはロールは、このようなアクセス許可を明示的に 1 つまたは複数持つ必要があります。例えば、IAM ユーザーまたはロールがアカウント 111122223333 内の S3 Access Grants インスタンスを呼び出して認証情報をリクエストできるように、`s3:GetDataAccess` アクセス許可を付与します。

このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
 "Version": "2012-10-17",
```

```
 "Statement": [
 {
  "Effect": "Allow",
  "Action": [
   "s3:GetDataAccess",
  ],
   "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
  }
 ]
}
```

IAM アイデンティティベースのポリシーの編集については、「AWS Identity and Access Management ガイド」の「IAM ポリシーの編集」を参照してください。

ステップ 4: アカウントの S3 Access Grants インスタンスで許可を作成して、2 番目のアカウントの IAM アイデンティティに S3 データの一部へのアクセスを許可します。

最後の設定ステップでは、アカウント 111122223333 の S3 Access Grants インスタンスに許可を作成します。これにより、2 番目のアカウント 444455556666 の IAM アイデンティティに、アカウント内の S3 データの一部へのアクセスが付与されます。これを行うには、Amazon S3 コンソール、CLI、API、SDK を使用できます。詳細については、「権限を作成する」を参照してください。

許可では、2 番目のアカウントの IAM アイデンティティの AWS ARN を指定して、アクセスを許可する S3 データの場所 (バケット、プレフィックス、またはオブジェクト) を指定します。この場所は、S3 Access Grants インスタンスに既に登録されている必要があります。詳細については、「ロケーションを登録する」を参照してください。必要に応じて、サブプレフィックスを指定することができます。例えば、アクセスを許可する場所がバケットで、そのバケット内の特定のオブジェクトへのアクセスをさらに制限したい場合は、オブジェクトキー名を `S3SubPrefix` フィールドで渡します。または、キー名が特定のプレフィックス (`2024-03-research-results/` など) で始まるバケット内のオブジェクトへのアクセスを制限したい場合は、`S3SubPrefix=2024-03-research-results/` を渡します。

2 番目のアカウントのアイデンティティに対するアクセス許可を作成するための CLI コマンドの例は、次のとおりです。詳細については、「権限を作成する」を参照してください。このコマンドの例を実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control create-access-grant \
--account-id 111122223333 \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix=prefixA* \
--permission READ \
```

```
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::444455556666:role/data-consumer-1
```

クロスアカウントアクセスを設定したら、2 番目のアカウントのユーザーまたはロールは次を実行できます。

- AWS RAM を介して共有されている S3 Access Grants インスタンスを一覧表示する `ListAccessGrantsInstances` を呼び出します。詳細については、「S3 Access Grants インスタンスの詳細を表示する」を参照してください。
- S3 Access Grants の一時的な認証情報をリクエストします。このようなリクエストを行う方法については、「S3 Access Grants を介して Amazon S3 データへのアクセスをリクエストする」を参照してください。

S3 Access Grants での AWS タグの使用

Amazon S3 Access Grants のタグは、Amazon S3 の オブジェクトタグ と類似の特性を持ちます。各タグはキーバリューのペアです。タグ付けできる S3 Access Grants のリソースは、S3 Access Grants インスタンス、ロケーション、権限 です。

> **Note**
>
> S3 アクセス権限でのタグ付けは、オブジェクトのタグ付けとは異なる API オペレーションを使用します。S3 Access Grants では、TagResource、UntagResource、ListTagsForResource API オペレーションを使用します。リソースは、S3 Access Grants インスタンス、登録済みロケーション、またはアクセス権限です。

オブジェクトタグ 同様、次の制限が適用されます。

- 新しい S3 Access Grants リソースにタグを追加したり、既存のリソースにタグを追加したりできます。
- 単一のリソースに関連付けることができるのは、最大 10 タグまでです。複数のタグを同じリソースに関連付ける場合は、一意のタグキーが必要となります。
- タグキーには最大 128 個の Unicode 文字、タグ値には最大 256 個の Unicode 文字を使用できます。Amazon S3 オブジェクトタグは、内部的に UTF-16 で表現されます。UTF-16 では、文字は 1 文字または 2 文字分を使用することに注意が必要です。

- キーバリューでは大文字と小文字が区別されます。

タグの制限については、「AWS Billing ユーザーガイド」の「ユーザー定義のタグの制限」を参照してください。

AWS Command Line Interface (AWS CLI)、Amazon S3 REST API、または AWS SDK を使用して S3 Access Grants のリソースにタグ付けできます。

AWS CLI の使用

AWS CLI をインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI をインストールする」を参照してください。

S3 Access Grants リソースは、作成時または作成後にタグ付けできます。S3 Access Grants のタグ付けまたはタグの削除方法の説明は、次の例のとおりです。登録済みの口ケーションとアクセスグラントに対しても同様のオペレーションを実行できます。

次のコマンド例を使用する際は、*user input placeholders* をユーザー自身の情報に置き換えます。

Example – タグ付けされた S3 Access Grants インスタンスを作成する

```
aws s3control create-access-grants-instance \
 --account-id 111122223333 \
 --profile access-grants-profile \
 --region us-east-2 \
 --tags Key=tagKey1,Value=tagValue1
```

レスポンス:

```
 {
    "CreatedAt": "2023-10-25T01:09:46.719000+00:00",
    "AccessGrantsInstanceId": "default",
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default"
}
```

Example – 既に作成済みの S3 Access Grants インスタンスにタグ付けする

```
aws s3control tag-resource \
--account-id 111122223333 \
```

```
--resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
--profile access-grants-profile \
--region us-east-2 \
--tags Key=tagKey2,Value=tagValue2
```

Example – S3 Access Grants インスタンスのタグを一覧表示する

```
aws s3control list-tags-for-resource \
--account-id 111122223333 \
--resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
--profile access-grants-profile \
--region us-east-2
```

レスポンス:

```
{
    "Tags": [
        {
            "Key": "tagKey1",
            "Value": "tagValue1"
        },
        {
            "Key": "tagKey2",
            "Value": "tagValue2"
        }
    ]
}
```

Example – S3 Access Grants インスタンスのタグ付けを解除する

```
aws s3control untag-resource \
 --account-id 111122223333 \
 --resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
 --profile access-grants-profile \
 --region us-east-2 \
 --tag-keys "tagKey2"
```

REST API の使用

Amazon S3 API を使用して、S3 Access Grants インスタンス、登録済みの口ケーション、またはアクセス権限のタグを付けたり、タグを解除したり、タグを一覧表示したりできます。REST API での

S3 Access Grants タグ管理のサポートの詳細については、「Amazon Simple Storage Service API リファレンス」の次のセクションを参照してください。

- TagResource
- UntagResource
- ListTagsForResource

S3 Access Grants の制約

S3 Access Grants には、次のとおりの制約があります。

> **Note**
>
> ユースケースがこのような制限を超える場合は、AWS サポート に連絡して制限の引き上げをリクエストしてください。

S3 Access Grants インスタンス

作成できるのは、アカウントにつき AWS リージョン ごとに単一の S3 Access Grants インスタンス です。「S3 Access Grants インスタンスを作成する」を参照してください。

S3 Access Grants ロケーション

S3 Access Grants インスタンスごとに登録できるのは、1,000 S3 Access Grants ロケーション です。「S3 Access Grants ロケーションを登録する」を参照してください。

権限

S3 Access Grants インスタンスごとに作成できるのは、100,000 権限 です。「権限を作成する」を参照してください。

S3 Access Grants の統合

S3 Access Grants は、次の AWS サービスと機能と連携できます。このページは、新しい統合が利用可能になると更新されます。

AWS IAM Identity Center

アプリケーション間での信頼されたアイデンティティのプロパゲーション

Amazon EMR

- S3 Access Grants を使用した Amazon EMR クラスターの起動

Amazon EMR on EKS

- S3 Access Grants を使用した Amazon EMR on EKS クラスターの起動

Amazon EMR Serverless アプリケーション

- S3 Access Grants を使用した Amazon EMR Serverless アプリケーションの起動

Amazon Athena

- IAM アイデンティティセンターが有効になっている Athena ワークグループの使用

ACL によるアクセス管理

アクセスコントロールリスト (ACL) はリソースベースのオプションの 1 つであり、バケットとオブジェクトへのアクセスを管理するために使用できます。ACL を使用して、基本的な読み取り/書き込み許可を他の AWS アカウントに付与できます。ACL によるアクセス許可の管理にはいくつかの制限があります。

例えば、アクセス許可を付与できるのは他の AWS アカウント だけで、自分のアカウントのユーザーには付与できません。条件付きアクセス許可を付与することはできず、アクセス許可を明示的に拒否することもできません。ACL が適しているのは、特定のいくつかのシナリオです。例えば、バケット所有者が他の AWS アカウント にオブジェクトのアップロードを許可している場合、これらのオブジェクトへのアクセス許可を管理する方法は、オブジェクトを所有する AWS アカウント によるオブジェクト ACL を使用する以外にありません。

S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、バケットにアップロードされる新しいオブジェクト所有権を制御し、ACL を無効にするのに使用できます。デフォルトでは、オブジェクト所有権はバケット所有者の強制設定に設定され、すべての ACL は無効になります。ACL を無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。

Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。ACL を無効にすると、誰がオブジェクトをバケットにアップロードしたかに関係なく、ポリシーを使用してバケット内のすべてのオブジェクトへのアクセスを制御できます。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

> **Important**
>
> バケットが S3 オブジェクト所有権のバケット所有者強制設定を使用している場合、ポリシーを使用してバケットとバケット中のオブジェクトへのアクセスを許可する必要があります。バケット所有者強制設定が有効になっている場合、アクセスコントロールリスト (ACL) の設定または ACL の更新は失敗し、`AccessControlListNotSupported` エラーコードが返されます。ACL の読み取り要求は引き続きサポートされています。

ACL の詳細については、次のトピックを参照してください。

トピック

アクセスコントロールリスト (ACL) の概要

Amazon S3 のアクセスコントロールリスト (ACL) では、バケットとオブジェクトへのアクセスを管理できます。各バケットとオブジェクトには、サブリソースとして ACL がアタッチされています。これにより、アクセスが許可される AWS アカウントまたはグループと、アクセスの種類が定義されます。リソースに対するリクエストを受け取ると、Amazon S3 は該当する ACL を確認して、リクエスタに必要なアクセス許可があることを確かめます。

S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、バケットにアップロードされる新しいオブジェクト所有権を制御し、ACL を無効にするのに使用できます。デフォルトでは、オブジェクト所有権はバケット所有者の強制設定に設定され、すべての ACL は無効になります。ACL を無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。

Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。ACL を無効にすると、誰がオブジェクトをバケットにアップロードしたかに関係なく、ポリシーを使用してバケット内のすべてのオブジェクトへのアクセスを制御できます。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

> **Important**
>
> バケットが S3 オブジェクト所有権のバケット所有者強制設定を使用している場合、ポリシーを使用してバケットとバケット中のオブジェクトへのアクセスを許可する必要があります。バケット所有者強制設定が有効になっている場合、アクセスコントロールリスト (ACL) の設定または ACL の更新は失敗し、`AccessControlListNotSupported` エラーコードが返されます。ACL の読み取り要求は引き続きサポートされています。

バケットまたはオブジェクトを作成すると、Amazon S3 はリソースのフルコントロールをリソースの所有者に付与するデフォルトの ACL を作成します。これを、以下のバケット ACL のサンプルで示します (デフォルトのオブジェクト ACL は同じ構造です)。

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>*** Owner-Canonical-User-ID ***</ID>
    <DisplayName>owner-display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="Canonical User">
        <ID>*** Owner-Canonical-User-ID ***</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

サンプル ACL には、`Owner` の正規ユーザー ID を通じて所有者を識別する AWS アカウント エレメントが含まれています。正規ユーザー ID を見つける手順については、「AWS アカウントの正規ユーザー ID の検索」を参照してください。`Grant` エレメントは、被付与者 (AWS アカウント またはあらかじめ定義されたグループ) と付与されたアクセス許可を識別します。このデフォルトの ACL には、所有者に対する 1 つの `Grant` エレメントがあります。`Grant` エレメントを追加してアクセス許可を付与します。各許可は被付与者とアクセス許可を識別します。

> **Note**
>
> 1 つの ACL には最大 100 個の許可を指定することができます。

トピック

被付与者とは

被付与者とは、AWS アカウントまたは事前定義済みのいずれかの Amazon S3 グループです。E メールアドレスまたは正規ユーザー ID を使用して AWS アカウントに許可を付与します。ただし、付与のリクエストでメールアドレスを指定すると、Amazon S3 はそのアカウントの正規ユーザー ID を確認して ACL に追加します。その結果、ACL には AWS アカウントの E メールアドレスではなく、常に AWS アカウントの正規ユーザー ID が含まれます。

アクセス権限を付与する場合は、各被付与者を *type*="*value*" のペアとして指定します。*type* は以下のいずれかです。

- `id` – 指定された値が AWS アカウントの正規ユーザー ID である場合
- `uri` - 事前定義されたグループにアクセス許可を付与する場合
- `emailAddress` – 指定された値が AWS アカウントの E メールアドレスである場合

> **Important**
>
> E メールアドレスを使用した被付与者の指定は、次の AWS リージョンでのみサポートされています。
>
> - 米国東部(バージニア北部)
> - 米国西部 (北カリフォルニア)
> - 米国西部 (オレゴン)

- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- 欧州 (アイルランド)
- 南米 (サンパウロ)

Amazon S3 でサポートされているリージョンとエンドポイントの一覧については、Amazon Web Services 全般のリファレンス の「リージョンとエンドポイント」を参照してください。

Example 例: E メールアドレス

例えば、次の x-amz-grant-read ヘッダーは、E メールアドレスによって識別される AWS アカウント に、オブジェクトデータとそのメタデータを読み取るアクセス許可を付与します。

```
x-amz-grant-read: emailAddress="xyz@example.com", emailAddress="abc@example.com"
```

Warning

他の AWS アカウントに自分のリソースへのアクセスを許可した場合、その AWS アカウントはアカウント内のユーザーに許可を委任できることに注意してください。これはクロスアカウントアクセスと呼ばれています。クロスアカウントアクセスの使用については、「IAM ユーザーガイド」の「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。

AWS アカウントの正規ユーザー ID の検索

正規ユーザー ID は、AWS アカウントに関連付けられています。この ID は、次のような長い文字列です。

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

アカウントの正規ユーザー ID を検索する方法については、「AWS Account Management リファレンスガイド」の「AWS アカウント の正規ユーザー ID を検索する」を参照してください。

また、AWS アカウントがアクセス許可を持つバケットまたはオブジェクトの ACL を読み取って、AWS アカウントの正規ユーザー ID を検索することもできます。許可リクエストによって個別の AWS アカウントに許可が付与された場合、ACL にはアカウントの正規ユーザー ID が含まれた許可エントリが追加されます。

> **Note**
>
> バケットをパブリックにした場合 (非推奨)、認証されていないどのユーザーもバケットにオブジェクトをアップロードできます。これらの匿名ユーザーは AWS アカウントを持っていません。匿名ユーザーがバケットにオブジェクトをアップロードすると、Amazon S3 によって特殊な正規ユーザー ID (65a011a29cdf8ec533ec3d1ccaae921c) がそのオブジェクトの所有者として ACL で追加されます。詳細については、「Amazon S3 のバケットとオブジェクトの所有権」を参照してください。

Amazon S3 の事前定義済みのグループ

Amazon S3 には、事前定義済みの一連のグループがあります。グループにアカウントアクセスを許可するときは、正規ユーザー ID の代わりに Amazon S3 のいずれかの URI を指定します。Amazon S3 には、以下の事前に定義されたグループが用意されています。

- Authenticated Users グループ – `http://acs.amazonaws.com/groups/global/AuthenticatedUsers` で表されます。

  このグループはすべて AWS アカウントを表しています。このグループへのアクセス許可により、AWS アカウント がリソースにアクセスできます。ただし、すべてのリクエストは署名(認証)されている必要があります。

  > **Warning**
  >
  > Authenticated Users グループにアクセスを許可すると、世界中の認証された AWS ユーザーがリソースにアクセスできます。

- All Users グループ – `http://acs.amazonaws.com/groups/global/AllUsers` で表されます。

  このグループへのアクセス許可により、世界中の誰でもリソースにアクセスすることが許可されます。リクエストは署名(認証)済み、または署名なし(匿名)とすることができます。署名なしのリクエストでは、リクエストの Authentication ヘッダーが省略されます。

> **Warning**
>
> All Users グループには、WRITE、WRITE_ACP、または FULL_CONTROL アクセス許可を一切付与しないことを強くお勧めします。例えば、WRITE のアクセス権限は所有者以外のユーザーが既存のオブジェクトを上書きまたは削除することを許可しませんが、WRITE のアクセス権限はすべてのユーザーがバケットにオブジェクトを格納することを許可し、これについてはお客様が請求されます。これらのアクセス許可の詳細については、次のセクション 付与できるアクセス許可 を参照してください。

- Log Delivery グループ – http://acs.amazonaws.com/groups/s3/LogDelivery で表されます。

  バケットに対する WRITE アクセス許可により、このグループはサーバーアクセスログ (「サーバーアクセスログによるリクエストのログ記録」を参照) をバケットに書き込むことができます。

> **Note**
>
> ACL を使用する場合、被付与者は AWS アカウントまたは事前定義済みのいずれかの Amazon S3 グループです。被付与者を IAM ユーザーとすることはできません。IAM 内の AWS ユーザーおよびアクセス許可の詳細については、「AWS Identity and Access Management の使用」を参照してください。

付与できるアクセス許可

以下の表に、Amazon S3 の ACL でサポートされている一連のアクセス許可を示します。ACL アクセス許可のセットは、オブジェクト ACL とバケット ACL で同じです。ただし、コンテキスト (バケット ACL かオブジェクト ACL か) に応じて、これらの ACL アクセス許可は特定のバケットまたはオブジェクトオペレーションのためのアクセス許可を付与します。この表では、アクセス許可の一覧と、オブジェクトとバケットにおけるその意味について説明しています。

Amazon S3 コンソールでの ACL アクセス権限の詳細については、「ACL の設定」を参照してください。

ACL アクセス許可

アクセス許可	バケット上で付与された場合	オブジェクト上で付与された場合
READ	被付与者がバケット内のオブジェクトをリストすることを許可します	被付与者がオブジェクトデータとそのメタデータを読み込むことを許可します
WRITE	被付与者がバケット内に新しいオブジェクトを作成できるようにします。既存のオブジェクトのバケット所有者およびオブジェクト所有者には、これらのオブジェクトの削除と上書きも許可します。	該当しません
READ_ACP	被付与者がバケット ACL を読み込むことを許可します	被付与者がオブジェクト ACL を読み込むことを許可します
WRITE_ACP	被付与者が該当するバケットの ACL を書き込むことを許可します	被付与者が該当するオブジェクトの ACL を書き込むことを許可します
FULL_CONTROL	バケットに対する READ、WRITE、READ_ACP、WRITE_AC のアクセス許可を被付与者に付与します	オブジェクトに対する READ、READ_ACP、WRITE_ACP のアクセス許可を被付与者に付与します

> **Warning**
>
> S3 バケットとオブジェクトにアクセス許可を付与するときは注意が必要です。例えば、あるバケットに対する WRITE のアクセス権を付与すると、被付与者はそのバケットにオブジェクトを作成できます。アクセス許可を付与する前に、「アクセスコントロールリスト (ACL) の概要」セクション全体を読むことを強くお勧めします。

ACL アクセス許可とアクセスポリシーのアクセス許可のマッピング

前の表に示したように、ACL で使用できるアクセス許可のセットは、アクセスポリシーで設定できるアクセス許可に比べると限定されています (「Amazon S3 のポリシーアクション」を参照してく

ださい)。これらのアクセス許可はそれぞれ、Amazon S3 の 1 つ以上のオペレーションを許可します。

次の表は、ACL アクセス権限のそれぞれが、対応するアクセスポリシーのアクセス許可にどのようにマッピングされるかを示します。ご覧のように、アクセスポリシーでは ACL よりも多くのアクセス許可が付与されています。ACL は主に、ファイルシステムのアクセス許可と同様に、基本的な読み取り/書き込みアクセス許可を付与するために使用されます。ACL の使用が適している場合の詳細については、Amazon S3 用 Identity and Access Management を参照してください。

Amazon S3 コンソールでの ACL アクセス権限の詳細については、「ACL の設定」を参照してください。

ACL アクセス許可	ACL アクセス許可がバケットに付与される場合の、対応するアクセスポリシーのアクセス許可	ACL アクセス許可がオブジェクトに付与される場合の、対応するアクセスポリシーのアクセス許可
READ	s3:ListBucket 、s3:ListBucketVersions 、および s3:ListBucketMultipartUploads	s3:GetObject および s3:GetObjectVersion
WRITE	s3:PutObject バケット所有者は、バケット内の任意のオブジェクトを作成、上書き、削除でき、オブジェクト所有者はそのオブジェクトに対する FULL_CONTROL を有します。 さらに、被付与者がバケット所有者であるときは、バケットの ACL で WRITE アクセス許可を付与すると、そのバケット内の任意のバージョンに対して s3:DeleteObjectVersion アクションを実行できるようになります。	該当しません

ACL アクセス許可	ACL アクセス許可がバケットに付与される場合の、対応するアクセスポリシーのアクセス許可	ACL アクセス許可がオブジェクトに付与される場合の、対応するアクセスポリシーのアクセス許可
READ_ACP	s3:GetBucketAcl	s3:GetObjectAcl および s3:GetObjectVersionAcl
WRITE_ACP	s3:PutBucketAcl	s3:PutObjectAcl および s3:PutObjectVersionAcl
FULL_CONTROL	これは、READ、WRITE、READ_ACP、および WRITE_ACP ACL アクセス許可を付与するのと同等です。したがって、この ACL アクセス許可は、対応するアクセスポリシーのアクセス許可の組み合わせにマッピングされます。	これは、READ、READ_ACP、および WRITE_ACP ACL アクセス許可を付与するのと同等です。したがって、この ACL アクセス許可は、対応するアクセスポリシーのアクセス許可の組み合わせにマッピングされます。

条件キー

アクセスポリシー権限を付与する場合、条件キーを使用して、バケットポリシーを使用するオブジェクトの ACL の値を制限できます。以下のコンテキストキーは ACL に対応しています。これらのコンテキストキーを使用して、リクエストで特定の ACL の使用を強制することができます。

- s3:x-amz-grant-read - 読み取りアクセスが必要です。
- s3:x-amz-grant-write - 書き込みアクセスが必要です。
- s3:x-amz-grant-read-acp - バケットの ACL への読み取りアクセスが必要です。
- s3:x-amz-grant-write-acp - バケットの ACL への書き込みアクセスが必要です。
- s3:x-amz-grant-full-control - フルコントロールが必要です。
- s3:x-amz-acl - 既定 ACL が必要です。

ACL 固有のヘッダーを含むポリシーの例については、「バケット所有者にフルコントロールを与えることを条件として s3:PutObject のアクセス許可を付与する」を参照してください。Amazon S3 固有の条件キーの完全なリストについては、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

一般的な Amazon S3 リクエストの **aclRequired** 値

承認に ACL を必要とした Amazon S3 リクエストを特定するには、Amazon S3 サーバーアクセスログまたは AWS CloudTrail の `aclRequired` 値を使用できます。CloudTrail または Amazon S3 サーバーアクセスログに表示される `aclRequired` 値は、呼び出されたオペレーションと、リクエスター、オブジェクト所有者、バケット所有者に関する特定の情報によって異なります。ACL が不要であったか、`bucket-owner-full-control` の既定 ACL を設定するか、リクエストがバケットポリシーで許可されている場合、`aclRequired` 値の文字列は Amazon S3 サーバーアクセスログでは「`-`」となり、CloudTrail では存在しません。

以下の表は、さまざまな Amazon S3 API オペレーションに対して CloudTrail または Amazon S3 サーバーアクセスログで期待される `aclRequired` 値を示しています。この情報から、どの Amazon S3 オペレーションが承認に関して ACL に依存しているかを理解できます。以下の表で、A、B、C は、リクエスター、オブジェクト所有者、バケット所有者と関連する各アカウントを表しています。アスタリスク (*) が付いているエントリは、A、B、C のうち、任意のアカウントを示します。

> **Note**
>
> 次の表の `PutObject` オペレーションは、特に明記しない限り、ACL を設定しないリクエストを示します。ただし、ACL が `bucket-owner-full-control` ACL である場合を除きます。`aclRequired` の値が NULL の場合、`aclRequired` は AWS CloudTrail ログに存在しないことを示します。

CloudTrail の **aclRequired** 値

オペレーション名	リクエスタ	オブジェクト所有者	バケット所有者	バケットポリシーはアクセスを許可する	**aclRequired** 値	理由
`GetObject`	A	A	A	はい/いいえ	null	同一アカウントアクセス
	A	B	A	はい/いいえ	null	同一アカウントアクセス (バケッ

オペレーション名	リクエスタ	オブジェクト所有者	バケット所有者	バケットポリシーはアクセスを許可する	**aclRequired** 値	理由
						ト所有者の強制による)
	A	A	B	あり	null	クロスアカウントアクセスがバケットポリシーで許可される
	A	A	B	なし	あり	クロスアカウントアクセスは ACL に依存する
	A	A	B	あり	null	クロスアカウントアクセスがバケットポリシーで許可される
	A	B	B	なし	あり	クロスアカウントアクセスは ACL に依存する

オペレーション名	リクエスタ	オブジェクト所有者	バケット所有者	バケットポリシーはアクセスを許可する	**aclRequired** 値	理由
	A	B	C	あり	null	クロスアカウントアクセスがバケットポリシーで許可される
	A	B	C	なし	あり	クロスアカウントアクセスは ACL に依存する
PutObject	A	該当しません	A	はい/いいえ	null	同一アカウントアクセス
	A	該当しません	B	あり	null	クロスアカウントアクセスがバケットポリシーで許可される
	A	該当しません	B	なし	あり	クロスアカウントアクセスは ACL に依存する

オペレーション名	リクエスタ	オブジェクト所有者	バケット所有者	バケットポリシーはアクセスを許可する	**aclRequired** 値	理由
ACL による PutObject (bucket-owner-full-control を除く)	*	該当しません	*	はい/いいえ	あり	リクエストは ACL を許可する
ListObjects	A	該当しません	A	はい/いいえ	null	同一アカウントアクセス
	A	該当しません	B	あり	null	クロスアカウントアクセスがバケットポリシーで許可される
	A	該当しません	B	なし	あり	クロスアカウントアクセスは ACL に依存する
DeleteObject	A	該当しません	A	はい/いいえ	null	同一アカウントアクセス

オペレーション名	リクエスタ	オブジェクト所有者	バケット所有者	バケットポリシーはアクセスを許可する	**aclRequired** 値	理由
	A	該当しません	B	あり	null	クロスアカウントアクセスがバケットポリシーで許可される
	A	該当しません	B	なし	あり	クロスアカウントアクセスは ACL に依存する
PutObjectAcl	*	*	*	はい/いいえ	あり	リクエストは ACL を許可する
PutBucketAcl	*	該当しません	*	はい/いいえ	あり	リクエストは ACL を許可する

> **Note**
>
> 次の表の REST.PUT.OBJECT オペレーションは、特に明記しない限り、ACL を設定しないリクエストを示します。ただし、ACL が bucket-owner-full-control ACL である場合を除きます。aclRequired 値の文字列「-」は、Amazon S3 サーバーアクセスログの NULL 値を示します。

Amazon S3 サーバーアクセスログの **aclRequired** 値

オペレーション名	リクエスタ	オブジェクト所有者	バケット所有者	バケットポリシーはアクセスを許可する	**aclRequired** 値	理由
REST.GET.OBJECT	A	A	A	はい/いいえ	-	同一アカウントアクセス
	A	B	A	はい/いいえ	-	同一アカウントアクセス (バケット所有者の強制による)
	A	A	B	あり	-	クロスアカウントアクセスがバケットポリシーで許可される
	A	A	B	なし	あり	クロスアカウントアクセスは ACL に依存する
	A	B	B	あり	-	クロスアカウントアクセスがバケットポリシーで許可される

オペレーション名	リクエスタ	オブジェクト所有者	バケット所有者	バケットポリシーはアクセスを許可する	**aclRequired** 値	理由
	A	B	B	なし	あり	クロスアカウントアクセスは ACL に依存する
	A	B	C	あり	-	クロスアカウントアクセスがバケットポリシーで許可される
	A	B	C	なし	あり	クロスアカウントアクセスは ACL に依存する
REST.PUT.OBJECT	A	該当しません	A	はい/いいえ	-	同一アカウントアクセス
	A	該当しません	B	あり	-	クロスアカウントアクセスがバケットポリシーで許可される

オペレーション名	リクエスタ	オブジェクト所有者	バケット所有者	バケットポリシーはアクセスを許可する	**aclRequired** 値	理由
	A	該当しません	B	なし	あり	クロスアカウントアクセスは ACL に依存する
ACL による REST.PUT.OBJECT (bucket-owner-full-control を除く)	*	該当しません	*	はい/いいえ	あり	リクエストは ACL を許可する
REST.GET.BUCKET	A	該当しません	A	はい/いいえ	-	同一アカウントアクセス
	A	該当しません	B	あり	-	クロスアカウントアクセスがバケットポリシーで許可される
	A	該当しません	B	なし	あり	クロスアカウントアクセスは ACL に依存する

オペレーション名	リクエスタ	オブジェクト所有者	バケット所有者	バケットポリシーはアクセスを許可する	**aclRequired** 値	理由
REST.DELETE.OBJECT	A	該当しません	A	はい/いいえ	-	同一アカウントアクセス
	A	該当しません	B	あり	-	クロスアカウントアクセスがバケットポリシーで許可される
	A	該当しません	B	なし	あり	クロスアカウントアクセスは ACL に依存する
REST.PUT.ACL	*	*	*	はい/いいえ	あり	リクエストは ACL を許可する

サンプル ACL

バケットの以下のサンプル ACL は、リソース所有者と一連の許可を識別します。形式は Amazon S3 REST API の ACL の XML 表現です。バケット所有者はリソースに対する FULL_CONTROL が許可されます。また、この ACL には、正規ユーザー ID で示されている 2 つの AWS アカウントと、前のセクションで説明した事前定義済みの 2 つの Amazon S3 グループに対して、リソースへの許可がどのように付与されるかが示されています。

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
```

```
 <Owner>
   <ID>Owner-canonical-user-ID</ID>
   <DisplayName>display-name</DisplayName>
 </Owner>
 <AccessControlList>
   <Grant>
     <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
       <ID>Owner-canonical-user-ID</ID>
       <DisplayName>display-name</DisplayName>
     </Grantee>
     <Permission>FULL_CONTROL</Permission>
   </Grant>

   <Grant>
     <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
       <ID>user1-canonical-user-ID</ID>
       <DisplayName>display-name</DisplayName>
     </Grantee>
     <Permission>WRITE</Permission>
   </Grant>

   <Grant>
     <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
       <ID>user2-canonical-user-ID</ID>
       <DisplayName>display-name</DisplayName>
     </Grantee>
     <Permission>READ</Permission>
   </Grant>

   <Grant>
     <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
       <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
     </Grantee>
     <Permission>READ</Permission>
   </Grant>
   <Grant>
     <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
       <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
     </Grantee>
     <Permission>WRITE</Permission>
   </Grant>
```

```
  </AccessControlList>
</AccessControlPolicy>
```

既定 ACL

Amazon S3 では、既定 ACL と呼ばれる事前定義済みの一連の許可がサポートされています。各既定 ACL には、あらかじめ定義された一連の被付与者とアクセス権限が含まれています。以下の表に、一連の既定 ACL と、関連するあらかじめ定義された許可を示します。

既定 ACL	適用先	ACL に追加されるアクセス許可
`private`	バケットとオブジェクト	所有者は `FULL_CONTROL` を取得します。他のユーザーにはアクセス許可は付与されません (デフォルト)。
`public-read`	バケットとオブジェクト	所有者は `FULL_CONTROL` を取得します。`AllUsers` グループ (被付与者とは を参照) は `READ` アクセス許可を取得します。
`public-read-write`	バケットとオブジェクト	所有者は `FULL_CONTROL` を取得します。`AllUsers` グループは `READ` および `WRITE` アクセス許可を取得します。通常、これをバケットで付与することはお勧めしません。
`aws-exec-read`	バケットとオブジェクト	所有者は `FULL_CONTROL` を取得します。Amazon EC2 には、Amazon S3 から Amazon マシンイメージ (AMI) バンドルを `GET` するための `READ` アクセスが許可されます。
`authenticated-read`	バケットとオブジェクト	所有者は `FULL_CONTROL` を取得します。`AuthenticatedUsers` グループは `READ` アクセス許可を取得します。
`bucket-owner-read`	オブジェクト	オブジェクト所有者は `FULL_CONTROL` を取得します。バケット所有者は `READ` を取得します。バケットの作成時にこの既定 ACL を指定しても、Amazon S3 には無視されます。

既定 ACL	適用先	ACL に追加されるアクセス許可
`bucket-owner-full-control`	オブジェクト	オブジェクト所有者とバケット所有者はオブジェクトに対する `FULL_CONTROL` を取得します。バケットの作成時にこの既定 ACL を指定しても、Amazon S3 には無視されます。
`log-delivery-write`	バケット	`LogDelivery` グループはバケットに対する `WRITE` および `READ_ACP` アクセス許可を取得します。ログの詳細については、サーバーアクセスログによるリクエストのログ記録 を参照してください。

> **Note**
>
> リクエストではこれらの既定 ACL を 1 つのみ指定できます。

`x-amz-acl` リクエストヘッダーを使用して、リクエストに既定 ACL を指定します。Amazon S3 が既定 ACL を含むリクエストを受信すると、あらかじめ定義された許可がリソースの ACL に追加されます。

ACL の設定

このセクションでは、アクセスコントロールリスト (ACL) を使用して S3 バケットとオブジェクトのアクセス許可を管理する方法について説明します。リソース ACL に許可を追加するには、AWS Management Console、AWS Command Line Interface (CLI)、REST API、または AWS SDK を使用します。

バケットに対するアクセス許可とオブジェクトに対するアクセス許可は相互に独立しています。オブジェクトはバケットからアクセス許可を継承しません。例えば、バケットを作成してユーザーに書き込みアクセスを許可した場合、そのユーザーから明示的にアクセスが許可されない限り、そのユーザーのオブジェクトにアクセスできません。

他の AWS アカウント ユーザーまたは事前定義されたグループに許可を与えることができます。アクセス許可が付与されたユーザーまたはグループは、被付与者と呼ばれます。デフォルトでは、バケットを作成した AWS アカウントである所有者が、完全な許可を持っています。

ユーザーまたはグループに付与する各アクセス許可により、バケットに関連付けられたエントリが ACL に追加されます。ACL は、被付与者と付与されたアクセス許可を識別するリストを表示します。

S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、バケットにアップロードされる新しいオブジェクト所有権を制御し、ACL を無効にするのに使用できます。デフォルトでは、オブジェクト所有権はバケット所有者の強制設定に設定され、すべての ACL は無効になります。ACL を無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。

Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。ACL を無効にすると、誰がオブジェクトをバケットにアップロードしたかに関係なく、ポリシーを使用してバケット内のすべてのオブジェクトへのアクセスを制御できます。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

> **Important**
>
> バケットが S3 オブジェクト所有権のバケット所有者強制設定を使用している場合、ポリシーを使用してバケットとバケット中のオブジェクトへのアクセスを許可する必要があります。バケット所有者強制設定が有効になっている場合、アクセスコントロールリスト (ACL) の設定または ACL の更新は失敗し、`AccessControlListNotSupported` エラーコードが返されます。ACL の読み取り要求は引き続きサポートされています。

> **Warning**
>
> すべてのユーザー (パブリックアクセス)または認証されたユーザーグループ (すべての AWS 認証ユーザー) のグループへの書き込みアクセスを許可しないことを強くお勧めします。これらのグループに書き込みアクセスを許可した場合の影響の詳細については、Amazon S3 の事前定義済みのグループ を参照してください。

S3 コンソールを使用した、バケットの ACL アクセス権限の設定

コンソールには、重複した被付与者のアクセス権限がまとめて表示されます。ACL の全リストを表示するには、Amazon S3 REST API、AWS CLI、または AWS SDK を使用します。

次の表は、Amazon S3 コンソールでバケットに設定できる ACL アクセス権限を示しています。

バケットの Amazon S3 コンソールの ACL アクセス権限

コンソールのアクセス権限	ACL アクセス許可	アクセス
オブジェクト – リスト	READ	被付与者がバケット内のオブジェクトをリストすることを許可します。
オブジェクト – 書き込み	WRITE	被付与者がバケット内に新しいオブジェクトを作成できるようにします。既存のオブジェクトのバケット所有者およびオブジェクト所有者については、これらのオブジェクトの削除と上書きも許可します。
バケット ACL – 読み取り	READ_ACP	被付与者がバケット ACL を読み込むことを許可します。
バケット ACL – 書き込み	WRITE_ACP	被付与者が該当するバケットの ACL を書き込むことを許可します。
全員 (パブリックアクセス): オブジェクト – リスト	READ	バケット内のオブジェクトに対するパブリック読み取りアクセス権を付与します。リストアクセス権を [Everyone (public access)] (全員 (パブリックアクセス)) に付与すると、世界中のユーザーがバケット内のオブジェクトにアクセスできます。
全員 (パブリックアクセス) : バケット ACL – 読み取り	READ_ACP	バケット ACL のパブリック読み取りアクセス権を付与します。[Everyone (public access)] (全員 (パブリックアクセス)) に読み取りアクセス権を付与すると、世界中のユーザーがバケット ACL にアクセスできます。

ACL アクセス権限の詳細については、アクセスコントロールリスト (ACL) の概要 を参照してください。

> **Important**
>
> バケットが S3 オブジェクト所有権のバケット所有者強制設定を使用している場合、ポリシーを使用してバケットとバケット中のオブジェクトへのアクセスを許可する必要がありま

す。バケット所有者強制設定が有効になっている場合、アクセスコントロールリスト (ACL) の設定または ACL の更新は失敗し、`AccessControlListNotSupported` エラーコードが返されます。ACL の読み取り要求は引き続きサポートされています。

バケットに ACL アクセス許可を設定する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets (バケット)] リストで、アクセス許可を設定するバケットの名前を選択します。
3. [Permissions] を選択します。
4. [アクセスコントロールリスト] で、[編集] を選択します。

   バケットに対する次の ACL アクセス許可を編集できます。

   オブジェクト

   - List (リスト) – 被付与者がバケット内のオブジェクトをリストすることを許可します
   - Write (書き込み) – 被付与者がバケット内に新しいオブジェクトを作成できるようにします。既存のオブジェクトのバケット所有者およびオブジェクト所有者については、これらのオブジェクトの削除と上書きも許可します。

     S3 コンソールでは、S3 ログデリバリーグループとバケット所有者 (AWS アカウント) にのみ書き込みアクセスを許可できます。他の被付与者には、書き込みアクセスを許可しないことを強くお勧めします。ただし、書き込みアクセスを許可する必要がある場合は、AWS CLI、AWS SDK、または REST API を使用できます。

   バケット ACL

   - Read (読み込み) – 被付与者がバケット ACL を読み込むことを許可します
   - Write (書き込み) – 被付与者が該当するバケットの ACL を書き込むことを許可します
5. バケット所有者の許可を変更するには、バケット所有者 (AWS アカウント)の横で、次の ACL アクセス権限を消去または選択します。

   - [Objects (オブジェクト)] – [List (リスト)] または [Write (書き込み)]
   - [Bucket ACL (バケット ACL)] – [Read (読み取り)] または [Write (書き込み)]

所有者とは、AWS アカウントのルートユーザーであり、AWS Identity and Access Management IAM ユーザーではありません。ルートユーザーの詳細については、IAM ユーザーガイドの「AWS アカウントのルートユーザー」を参照してください。

6. 一般ユーザー (インターネット上のすべてのユーザー) のアクセス許可を付与または取り消すには、[Everyone (public access) (全員 (パブリックアクセス))] の横で、次の ACL アクセス許可をクリアまたは選択します。

   - [Objects (オブジェクト)] – [List (リスト)]
   - [Bucket ACL (バケット ACL)] – [Read (読み取り)]

   > ⚠ Warning
   >
   > [全員] グループに S3 バケットへのパブリックアクセスを付与するときは注意が必要です。このグループにアクセスを付与すると、世界中のすべてのユーザーがバケットにアクセスできます。種類にかかわらず、S3 バケットへのパブリック書き込みアクセスは一切付与しないことを強くお勧めします。

7. AWS アカウント があれば誰でも許可を付与するか取り消すために、認証済みユーザーグループ (AWS アカウント があれば誰でも可) の横で、次の ACL アクセス権限を消去または選択します。

   - [Objects (オブジェクト)] – [List (リスト)]
   - [Bucket ACL (バケット ACL)] – [Read (読み取り)]

8. Amazon S3 がバケットにサーバーアクセスログを書き込むアクセス許可を付与または取り消すには、[S3 log delivery group (S3 ログ配信グループ)] で、次の ACL アクセス許可をクリアまたは選択します。

   - [Objects (オブジェクト)] – [List (リスト)] または [Write (書き込み)]
   - [Bucket ACL (バケット ACL)] – [Read (読み取り)] または [Write (書き込み)]

   バケットがアクセスログを受け取るターゲットバケットとして設定されている場合、バケットのアクセス許可は [ログ配信] グループのバケットへの書き込みアクセスを許可する必要があります。バケット上のサーバアクセスログ記録を有効にすると、Amazon S3 コンソールは、[Log Delivery (ログ配信)] グループにログを受信することを選択したターゲットバケットへの

書き込みアクセス権を付与します。サーバーアクセスログ記録の詳細については、「Amazon S3 サーバーアクセスログを有効にします。」を参照してください。

9. 別の AWS アカウントへの許可を付与するには、次の手順を実行します。

   a. [Add grantee (被付与者の追加)] を選択します。

   b. [Grantee] (被付与者) ボックスに、他の AWS アカウント の正規 ID を入力します。

   c. 次の ACL アクセス許可から選択します。

      - [Objects (オブジェクト)] – [List (リスト)] または [Write (書き込み)]
      - [Bucket ACL (バケット ACL)] – [Read (読み取り)] または [Write (書き込み)]

> ⚠ Warning
>
> 他の AWS アカウントに自分のリソースへのアクセスを許可した場合、その AWS アカウントはアカウント内のユーザーに許可を委任できることに注意してください。これはクロスアカウントアクセスと呼ばれています。クロスアカウントアクセスの使用については、「IAM ユーザーガイド」の「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。

10. 別の AWS アカウント へのアクセスを削除するには、他の AWS アカウント へのアクセスで、削除を選択します。

11. 変更を保存するには、[変更の保存] を選択します。

S3 コンソールを使用した、オブジェクトの ACL アクセス権限の設定

コンソールには、重複した被付与者のアクセス権限がまとめて表示されます。ACL の全リストを表示するには、Amazon S3 REST API、AWS CLI、または AWS SDK を使用します。次の表に、Amazon S3 コンソールでオブジェクト用に設定できる ACL アクセス権限を示します。

オブジェクト用の Amazon S3 コンソールの ACL アクセス権限

コンソールのアクセス権限	ACL アクセス許可	アクセス
オブジェクト – 読み取り	READ	被付与者がオブジェクトデータとそのメタデータを読み込むことを許可します。

コンソールのアクセス権限	ACL アクセス許可	アクセス
オブジェクト ACL – 読み取り	READ_ACP	被付与者がオブジェクト ACL を読み込むことを許可します。
オブジェクト ACL – 書き込み	WRITE_ACP	被付与者が該当するオブジェクトの ACL を書き込むことを許可します

ACL アクセス権限の詳細については、アクセスコントロールリスト (ACL) の概要 を参照してください。

> **Important**
>
> バケットが S3 オブジェクト所有権のバケット所有者強制設定を使用している場合、ポリシーを使用してバケットとバケット中のオブジェクトへのアクセスを許可する必要があります。バケット所有者強制設定が有効になっている場合、アクセスコントロールリスト (ACL) の設定または ACL の更新は失敗し、AccessControlListNotSupported エラーコードが返されます。ACL の読み取り要求は引き続きサポートされています。

オブジェクトに ACL アクセス権限を設定する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets] (バケット) リストで、オブジェクトが含まれるバケットの名前を選択します。
3. オブジェクトのリストで、アクセス許可を設定するオブジェクトの名前を選択します。
4. [Permissions] を選択します。
5. [アクセスコントロールリスト (ACL)] で、[編集] を選択します。

   オブジェクトに対する次の ACL アクセス許可を編集できます。

オブジェクト

- Read (読み込み) – 被付与者がオブジェクトデータとそのメタデータを読み込むことを許可します

オブジェクト ACL

- 読み込み (Read) – 被付与者がオブジェクト ACL を読み込むことを許可します
- Write (書き込み) – 被付与者が該当するオブジェクトの ACL を書き込むことを許可します S3 コンソールでは、バケット所有者 (AWS アカウント) にのみ書き込みアクセスを許可できます。他の被付与者には、書き込みアクセスを許可しないことを強くお勧めします。ただし、書き込みアクセスを許可する必要がある場合は、AWS CLI、AWS SDK、または REST API を使用できます。

6. 以下のオブジェクトのアクセス許可を管理できます。

   a. オブジェクト所有者へのアクセス

      所有者とは、AWS アカウントのルートユーザーであり、AWS Identity and Access Management IAM ユーザーではありません。ルートユーザーの詳細については、IAM ユーザーガイドの「AWS アカウントのルートユーザー」を参照してください。

      所有者のオブジェクトアクセス許可を変更するには、オブジェクト所有者へのアクセスで、お客様の AWS アカウント (所有者) を選択します。

      変更するアクセス許可のチェックボックスをオンにして、[保存] を選択します。

   b. その他の AWS アカウントのアクセス

      別の AWS から AWS アカウント ユーザーに許可を与えるには、他の AWS アカウント へのアクセスで、アカウントの追加を選択します。[Enter an ID] (ID の入力) フィールドに、オブジェクトのアクセス許可を付与する AWS ユーザーの正規 ID を入力します。正規 ID の検索に関する詳細は、Amazon Web Services 全般のリファレンス の「AWS アカウント 識別子」を参照してください。最大 99 人のユーザーを追加できます。

      ユーザーに付与するアクセス許可のチェックボックスをオンにして、[保存] を選択します。アクセス許可についての情報を表示するには、ヘルプアイコンを選択します。

c. パブリックアクセス

オブジェクトへのアクセスを一般のユーザー (世界中のすべてのユーザー) に許可するには、[パブリックアクセス] で [全員] を選択します。パブリックアクセス許可を付与すると、世界中の誰でもオブジェクトにアクセスできるようになります。

付与するアクセス許可のチェックボックスをオンにして、[保存] を選択します。

> **Warning**
>
> - [Everyone (全員)] グループに Amazon S3 オブジェクトへの匿名アクセスを付与するときは注意が必要です。このグループにアクセスを付与すると、世界中のすべてのユーザーがオブジェクトにアクセスできます。すべてのユーザーにアクセスを付与する必要がある場合は、[Read objects (オブジェクトの読み取り)] のみのアクセス許可を付与することを強くお勧めします。
> - [全員] グループにオブジェクト書き込み許可は付与しないことを強くお勧めします。付与すると、すべてのユーザーに対して、オブジェクトの ACL アクセス許可を上書きすることが許可されます。

AWS SDK の使用

このセクションでは、バケットやオブジェクトでアクセスコントロールリスト (ACL) の付与を設定する方法の例を示します。

> **Important**
>
> バケットが S3 オブジェクト所有権のバケット所有者強制設定を使用している場合、ポリシーを使用してバケットとバケット中のオブジェクトへのアクセスを許可する必要があります。バケット所有者強制設定が有効になっている場合、アクセスコントロールリスト (ACL) の設定または ACL の更新は失敗し、`AccessControlListNotSupported` エラーコードが返されます。ACL の読み取り要求は引き続きサポートされています。

Java

このセクションでは、バケットやオブジェクトでアクセスコントロールリスト (ACL) の付与を設定する方法の例を示します。最初の例では、既定 ACL を使用するバケッ

ト (「既定 ACL」を参照) を作成します。次に、カスタムアクセス許可の付与のリスト を作成して、既定の ACL をカスタム付与を含む ACL で置き換えます。2 番目の例では、AccessControlList.grantPermission( ) メソッドを使用して ACL を変更する方法を示します。

Example バケットを作成し、S3 ログ配信グループにアクセス許可を付与する既定 ACL を指定する

この例では、バケットを作成します。リクエストで既定の ACL を指定し、バケットにログを書き込むアクセス許可をログ配信グループに付与します。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.ArrayList;

public class CreateBucketWithACL {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String userEmailForReadPermission = "*** user@example.com ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .build();

            // Create a bucket with a canned ACL. This ACL will be replaced by the
            // setBucketAcl()
            // calls below. It is included here for demonstration purposes.
            CreateBucketRequest createBucketRequest = new
 CreateBucketRequest(bucketName, clientRegion.getName())
                    .withCannedAcl(CannedAccessControlList.LogDeliveryWrite);
            s3Client.createBucket(createBucketRequest);

            // Create a collection of grants to add to the bucket.
```

```
            ArrayList<Grant> grantCollection = new ArrayList<Grant>();

            // Grant the account owner full control.
            Grant grant1 = new Grant(new
 CanonicalGrantee(s3Client.getS3AccountOwner().getId()),
                    Permission.FullControl);
            grantCollection.add(grant1);

            // Grant the LogDelivery group permission to write to the bucket.
            Grant grant2 = new Grant(GroupGrantee.LogDelivery, Permission.Write);
            grantCollection.add(grant2);

            // Save grants by replacing all current ACL grants with the two we just
 created.
            AccessControlList bucketAcl = new AccessControlList();
            bucketAcl.grantAllPermissions(grantCollection.toArray(new Grant[0]));
            s3Client.setBucketAcl(bucketName, bucketAcl);

            // Retrieve the bucket's ACL, add another grant, and then save the new
 ACL.
            AccessControlList newBucketAcl = s3Client.getBucketAcl(bucketName);
            Grant grant3 = new Grant(new
 EmailAddressGrantee(userEmailForReadPermission), Permission.Read);
            newBucketAcl.grantAllPermissions(grant3);
            s3Client.setBucketAcl(bucketName, newBucketAcl);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example 既存のオブジェクトの ACL を更新する

この例では、オブジェクトの ACL を更新します。この例では次のタスクを実行しています。

- オブジェクトの ACL を取得する

- すべての既存のアクセス許可を削除して ACL をクリアする
- 2 つのアクセス許可として、所有者へのフルアクセスと、メールアドレスで識別されたユーザーへの WRITE_ACP (付与できるアクセス許可 を参照) を追加する
- ACL をオブジェクトに保存する

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.CanonicalGrantee;
import com.amazonaws.services.s3.model.EmailAddressGrantee;
import com.amazonaws.services.s3.model.Permission;

import java.io.IOException;

public class ModifyACLExistingObject {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String emailGrantee = "*** user@example.com ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Get the existing object ACL that we want to modify.
            AccessControlList acl = s3Client.getObjectAcl(bucketName, keyName);

            // Clear the existing list of grants.
            acl.getGrantsAsList().clear();

            // Grant a sample set of permissions, using the existing ACL owner for
 Full
```

```
            // Control permissions.
            acl.grantPermission(new CanonicalGrantee(acl.getOwner().getId()),
 Permission.FullControl);
            acl.grantPermission(new EmailAddressGrantee(emailGrantee),
 Permission.WriteAcp);

            // Save the modified ACL back to the object.
            s3Client.setObjectAcl(bucketName, keyName, acl);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

Example バケットを作成し、S3 ログ配信グループにアクセス許可を付与する既定 ACL を指定する

この C# の例では、バケットを作成します。リクエストで既定の ACL を指定し、バケットにログを書き込むアクセス許可をログ配信グループに付与します。

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ManagingBucketACLTest
    {
        private const string newBucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
```

```
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            CreateBucketUseCannedACLAsync().Wait();
        }

        private static async Task CreateBucketUseCannedACLAsync()
        {
            try
            {
                // Add bucket (specify canned ACL).
                PutBucketRequest putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = S3Region.EUW1, // S3Region.US,
                                                  // Add canned ACL.
                    CannedACL = S3CannedACL.LogDeliveryWrite
                };
                PutBucketResponse putBucketResponse = await
client.PutBucketAsync(putBucketRequest);

                // Retrieve bucket ACL.
                GetACLResponse getACLResponse = await client.GetACLAsync(new
GetACLRequest
                {
                    BucketName = newBucketName
                });
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("S3 error occurred. Exception: " +
amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
```

```
}
```

Example 既存のオブジェクトの ACL を更新する

この C# の例では、既存のオブジェクトで ACL を更新します。この例では次のタスクを実行しています。

- オブジェクトの ACL を取得する。
- すべての既存のアクセス許可を削除して ACL をクリアする。
- 2 つのアクセス許可として、所有者へのフルアクセスと、メールアドレスで識別されたユーザーへの WRITE_ACP を追加する。
- `PutAcl` リクエストを送信して ACL を保存する。

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ManagingObjectACLTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key name ***";
        private const string emailAddress = "*** email address ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            TestObjectACLTestAsync().Wait();
        }
        private static async Task TestObjectACLTestAsync()
        {
```

```
            try
            {
                    // Retrieve the ACL for the object.
                    GetACLResponse aclResponse = await client.GetACLAsync(new
GetACLRequest
                    {
                        BucketName = bucketName,
                        Key = keyName
                    });

                    S3AccessControlList acl = aclResponse.AccessControlList;

                    // Retrieve the owner (we use this to re-add permissions after
we clear the ACL).
                    Owner owner = acl.Owner;

                    // Clear existing grants.
                    acl.Grants.Clear();

                    // Add a grant to reset the owner's full permission (the
previous clear statement removed all permissions).
                    S3Grant fullControlGrant = new S3Grant
                    {
                        Grantee = new S3Grantee { CanonicalUser = owner.Id },
                        Permission = S3Permission.FULL_CONTROL

                    };

                    // Describe the grant for the permission using an email address.
                    S3Grant grantUsingEmail = new S3Grant
                    {
                        Grantee = new S3Grantee { EmailAddress = emailAddress },
                        Permission = S3Permission.WRITE_ACP
                    };
                    acl.Grants.AddRange(new List<S3Grant> { fullControlGrant,
grantUsingEmail });

                    // Set a new ACL.
                    PutACLResponse response = await client.PutACLAsync(new
PutACLRequest
                    {
                        BucketName = bucketName,
                        Key = keyName,
                        AccessControlList = acl
```

```
                    });
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

REST API の使用

Amazon S3 API を使用して、バケットまたはオブジェクトの作成時に、ACL を設定できます。Amazon S3 では、既存のバケットまたはオブジェクトに ACL を設定する API も提供します。これらの API は ACL を設定する次のメソッドを提供します。

- リクエストヘッダーを使用した ACL の設定 – リソース (バケットまたはオブジェクト) を作成するリクエストを送るときに、リクエストヘッダーを使用して ACL を設定します。これらのヘッダーを使用して、既定 ACL を指定するか、許可を明示的に指定 (被付与者とアクセス許可を明示的に識別) します。
- リクエストボディを使用した ACL の設定 – 既存のリソースに ACL を設定するリクエストを送信するときに、リクエストのヘッダーまたはボディを使用して ACL を設定できます。

REST API での ACL の管理のサポートの詳細については、Amazon Simple Storage Service API リファレンスの以下のセクションを参照してください。

- GET Bucket acl
- PUT Bucket acl
- GET Object acl
- PUT Object acl
- PUT Object
- PUT Bucket

- PUT Object - Copy
- Initiate Multipart Upload

> **Important**
>
> バケットが S3 オブジェクト所有権のバケット所有者強制設定を使用している場合、ポリシーを使用してバケットとバケット中のオブジェクトへのアクセスを許可する必要があります。バケット所有者強制設定が有効になっている場合、アクセスコントロールリスト (ACL) の設定または ACL の更新は失敗し、`AccessControlListNotSupported` エラーコードが返されます。ACL の読み取り要求は引き続きサポートされています。

アクセスコントロールリスト (ACL) 固有の要求ヘッダー

ヘッダーを使用して、アクセスコントロールリスト (ACL) ベースのアクセス許可を付与できます。デフォルトでは、すべてのオブジェクトがプライベートです。所有者だけがフルアクセスコントロールを持っています。新しいオブジェクトを追加するときに、個々の AWS アカウントに、または Amazon S3 であらかじめ定義されたグループに許可を付与できます。これらのアクセス許可は、オブジェクトのアクセスコントロールリスト (ACL) に追加されます。詳細については、「アクセスコントロールリスト (ACL) の概要」を参照してください。

この操作では、次の 2 つの方法のいずれかを使用してアクセス許可を付与できます。

- 既定 ACL (**`x-amz-acl`**) — Amazon S3 は、既定 ACL と呼ばれる、あらかじめ定義された一連の ACL をサポートしています。各既定 ACL には、あらかじめ定義された一連の被付与者とアクセス権限が含まれています。詳細については、「既定 ACL」を参照してください。
- アクセス許可 — 特定の AWS アカウント またはグループにアクセス許可を明示的に付与するには、次のヘッダーを使用します。各ヘッダーは、Amazon S3 が ACL でサポートする特定のアクセス許可にマッピングされます。詳細については、「アクセスコントロールリスト (ACL) の概要」を参照してください。ヘッダーでは、特定のアクセス許可を取得する被付与者のリストを指定します。
  - x-amz-grant-read
  - x-amz-grant-write
  - x-amz-grant-read-acp
  - x-amz-grant-write-acp
  - x-amz-grant-full-control

AWS CLI の使用

AWS CLI を使用した ACL の管理の詳細については、AWS CLI Command Reference の put-bucket-acl を参照してください。

> **Important**
>
> バケットが S3 オブジェクト所有権のバケット所有者強制設定を使用している場合、ポリシーを使用してバケットとバケット中のオブジェクトへのアクセスを許可する必要があります。バケット所有者強制設定が有効になっている場合、アクセスコントロールリスト (ACL) の設定または ACL の更新は失敗し、`AccessControlListNotSupported` エラーコードが返されます。ACL の読み取り要求は引き続きサポートされています。

ACL のポリシーの例

バケットポリシーで条件キーを使用して、Amazon S3 へのアクセスをコントロールできます。

トピック

- バケット所有者にフルコントロールを与えることを条件として s3:PutObject のアクセス許可を付与する
- x-amz-acl ヘッダーの条件を使用した s3:PutObject アクセス許可の付与

バケット所有者にフルコントロールを与えることを条件として s3:PutObject のアクセス許可を付与する

PutObject オペレーションでは、アクセスコントロールリスト (ACL) に固有のヘッダーを使用して、ACL に基づいてアクセス許可を付与することができます。バケット所有者は、これらのキーを使用して条件を設定し、ユーザーがオブジェクトをアップロードする場合に特定のアクセス許可を要求することができます。

例えば、アカウント A がバケット所有者であり、アカウント管理者がアカウント B のユーザー Dave に対して、オブジェクトをアップロードするアクセス許可を付与するとします。デフォルトでは、Dave がアップロードするオブジェクトはアカウント B に所有されるため、アカウント A にはそれらのオブジェクトに対するアクセス許可は付与されません。ただし、バケット所有者は請求書を支払うために、Dave がアップロードするオブジェクトに対する完全なアクセス許可を必要としています。この対処方法として、アカウント A の管理者は、明示的に完全なアクセス許可を付与

するか、既定 ACL を使用する ACL 固有のヘッダーをリクエストに含むことを条件として、Dave に s3:PutObject のアクセス許可を付与できます。詳細については、「PutObject」を参照してください。

x-amz-full-control ヘッダーを必須にする

リクエストで、バケット所有者へのフルコントロールのアクセス許可が含まれる x-amz-full-control ヘッダーを要求できます。以下のバケットポリシーは、s3:PutObject 条件キーを使用して、リクエストに s3:x-amz-grant-full-control ヘッダーを含めることを条件とする x-amz-full-control のアクセス許可をユーザー Dave に付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountB-ID:user/Dave"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::awsexamplebucket1/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
        }
      }
    }
  ]
}
```

> **Note**
>
> この例では、クロスアカウントのアクセス許可を付与しています。ただし、許可を付与される Dave がバケットを所有している AWS アカウントに属している場合、この条件付き許可は不要になります。これは、ユーザーがアップロードするオブジェクトは Dave が属する親アカウントによって所有されるためです。

明示的な拒否を追加する

前述のバケットポリシーでは、アカウント B のユーザー Dave に条件付きのアクセス許可が付与されます。ただし、このポリシーが有効であっても、Dave が他のポリシーに従って、条件なしで同じアクセス許可を取得する場合があります。例えば、Dave が属するグループに、条件なしで s3:PutObject のアクセス許可が付与される場合があります。このようなアクセス許可の抜け穴を避けるには、明示的な拒否を追加して、より厳格なアクセスポリシーを記述する必要があります。次の例では、バケット所有者に完全なアクセス許可を付与するヘッダーをリクエストに含めない場合、Dave に対してアップロードのアクセス許可を明示的に拒否します。明示的な拒否は、付与されている他のアクセス許可に常に優先されます。以下は、明示的な拒否が追加された、改訂済みアクセスポリシーの例です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        }
    ]
```

```
}
```

AWS CLI でポリシーをテストする

2 つの AWS アカウントがある場合は、AWS Command Line Interface (AWS CLI) を使用してポリシーをテストできます。ポリシーをアタッチしたら、Dave の認証情報を使用し、次の AWS CLI put-object コマンドを実行してアクセス許可をテストします。Dave の認証情報は、--profile パラメータを追加して指定します。バケット所有者にフルコントロールのアクセス許可を付与するには、--grant-full-control パラメータを追加します。AWS CLI のセットアップおよび使用の詳細については、AWS CLI を使用した Amazon S3 での開発 を参照してください。

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg
 --grant-full-control id="AccountA-CanonicalUserID" --profile AccountBUserProfile
```

x-amz-acl ヘッダーを必須にする

バケット所有者にフルコントロールのアクセス許可を付与する既定 ACL が指定された x-amz-acl ヘッダーを要求できます。リクエストに x-amz-acl ヘッダーを含めることを義務付けるには、以下の例のように Condition ブロックのキーと値のペアを置き換え、s3:x-amz-acl 条件キーを指定します。

```
"Condition": {
    "StringEquals": {
        "s3:x-amz-acl": "bucket-owner-full-control"
    }
}
```

AWS CLI を使用してアクセス許可をテストするには、--acl パラメータを指定します。これにより、AWS CLI は送信するリクエストに x-amz-acl ヘッダーを追加します。

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg
 --acl "bucket-owner-full-control" --profile AccountBadmin
```

x-amz-acl ヘッダーの条件を使用した s3:PutObject アクセス許可の付与

以下のバケットポリシーは、オブジェクトをパブリックに読み取り可能にする x-amz-acl ヘッダーがリクエストに含まれている場合に、2 つの AWS アカウントに s3:PutObject の許可を付与します。Condition ブロックでは、StringEquals 条件を使用し、キーと値のペアとして "s3:x-amz-acl":["public-read"] を評価に使用できます。このキーと値のペアで、s3:x-amz-acl は、「s3:」というプレフィックスが示すとおり Amazon S3 固有のキーです。

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
         "Sid":"AddCannedAcl",
            "Effect":"Allow",
         "Principal": {
                "AWS": [
                "arn:aws:iam::Account1-ID:root",
    "arn:aws:iam::Account2-ID:root"
    ]
            },
         "Action":"s3:PutObject",
            "Resource": ["arn:aws:s3:::awsexamplebucket1/*"],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl":["public-read"]
                }
            }
        }
    ]
}
```

> **Important**
>
> すべての条件が、すべてのアクションに対して意味を成すわけではありません。例えば、Amazon S3 の `s3:CreateBucket` のアクセス許可を付与するポリシーに条件として `s3:LocationConstraint` を含めることは理にかなっています。ただし、この条件を `s3:GetObject` のアクセス許可を付与するポリシーに含めることは意味がありません。Amazon S3 では、このような Amazon S3 固有の条件を含むセマンティックエラーをテストすることができます。ただし、IAM ユーザーまたはロールのポリシーを作成し、意味的に無効な Amazon S3 条件を含めても、IAM は Amazon S3 条件を検証できないため、エラーは報告されません。

Amazon S3 ストレージへのパブリックアクセスのブロック

Amazon S3 のパブリックアクセスブロック機能は、Amazon S3 のリソースへのパブリックアクセスの管理に役立つ、アクセスポイント、バケット、アカウントの設定を提供します。デフォルトでは、新しいバケット、アクセスポイント、およびオブジェクトはパブリックアクセスを許可しません。た

だし、ユーザーはバケットポリシー、アクセスポイントポリシー、またはオブジェクトのアクセス許可を変更することで、パブリックアクセスを許可できます。S3 ブロックパブリックアクセス設定は、これらのポリシーやアクセス許可を上書きして、これらのリソースへのパブリックアクセスを制限できるようにします。

S3 のパブリックアクセスブロックでは、アカウント管理者やバケット所有者は Amazon S3 のリソースへのパブリックアクセスを制限する一元的な管理を簡単に設定して、リソースがどのように作成されたかに関係なく強制的に適用することができます。

パブリックブロックアクセスを設定する手順については、パブリックアクセスブロックの設定 を参照してください。

Amazon S3 はバケットやオブジェクトへのアクセスのリクエストを受け取ると、バケットやバケット所有者のアカウントに適用されているパブリックアクセスブロック設定があるかどうかを確認します。リクエストがアクセスポイントを経由している場合、Amazon S3 はアクセスポイントのパブリックアクセスブロック設定も確認します。リクエストされたアクセスを禁止する既存のパブリックアクセスブロック設定がある場合、Amazon S3 はそのリクエストを拒否します。

Amazon S3 のパブリックアクセスブロックには、4 つの設定があります。これらの設定は、独立しており、任意の組み合わせで使用できます。各設定は、アクセスポイント、バケット、または AWS アカウント全体に適用できます。アクセスポイント、バケット、アカウントのパブリックアクセスブロック設定が異なる場合、Amazon S3 は、アクセスポイント、バケット、アカウントの設定の組み合わせで最も制限が厳しいものを適用します。

Amazon S3 は、パブリックアクセスブロック設定でオペレーションが禁止されているかどうかを評価し、アクセスポイント、バケット、アカウントの設定に違反しているすべてのリクエストを拒否します。

> **Important**
>
> パブリックアクセスは、アクセスコントロールリスト (ACL)、アクセスポイントポリシー、バケットポリシー、またはそのすべてからバケットおよびオブジェクトに付与されます。Amazon S3 のすべてのアクセスポイント、バケット、オブジェクトへのパブリックアクセスを確実にブロックするために、アカウントへのパブリックアクセスをブロックする 4 つの設定をすべて有効にすることをお勧めします。これらの設定によって、現在および将来のバケットおよびアクセスポイントのパブリックアクセスはすべてブロックされます。
> これらの設定を適用する前に、アプリケーションがパブリックアクセスなしで正しく動作することを確認してください。「Amazon S3 を使用して静的ウェブサイトをホスティングする」に示す静的なウェブサイトをホストする場合など、バケットやオブジェクトにある程度

のパブリックアクセスが必要な場合は、ストレージのユースケースに合わせて個別に設定をカスタマイズできます。
[パブリックアクセスをブロック] を有効にすると、S3 リソースに直接アタッチされているリソースポリシーやアクセスコントロールリスト (ACL) を通じてパブリックアクセスが付与されるのを防ぐことができるため、リソースを保護できます。[パブリックアクセスをブロック] を有効にすること以外にも、次のポリシーを慎重に検査して、パブリックアクセスを付与していないことを確認します。

- 関連する AWS プリンシパル (IAM ロールなど) にアタッチされているアイデンティティベースのポリシー
- 関連する AWS リソースにアタッチされたリソースベースのポリシー (AWS Key Management Service (KMS) キーなど)

Note

- パブリックアクセスブロック設定は、アクセスポイント、バケット、AWS アカウントに対してのみ有効にすることができます。Amazon S3 では、オブジェクトごとのパブリックアクセスブロック設定はサポートされていません。
- アカウントにブロックパブリックアクセス設定を適用すると、その設定はすべての AWS リージョンにグローバルに適用されます。設定はすべてのリージョンで即時または同時に有効になるわけではありませんが、最終的にはすべてのリージョンに反映されます。

トピック

- パブリックアクセスブロック設定
- アクセスポイントでのパブリックアクセスブロックオペレーションの実行
- 「パブリック」の意味
- IAM Access Analyzer for S3 を使用したパブリックバケットの確認
- アクセス許可
- パブリックアクセスブロックの設定
- アカウントのパブリックアクセスブロック設定の構成
- S3 バケットへのパブリックアクセスブロック設定の構成

パブリックアクセスブロック設定

S3 のブロックパブリックアクセスには 4 つの設定があります。これらの設定は、個別のアクセスポイント、バケット、または AWS アカウント 全体に任意の組み合わせで適用できます。設定をアカウントに適用すると、その設定はアカウントが所有するすべてのバケットとアクセスポイントに適用されます。同様に、設定をバケットに適用すると、その設定はバケットに関連付けられているすべてのアクセスポイントに適用されます。

次の表に、利用可能な設定が含まれます。

名前	説明
`BlockPublicAcls`	このオプションを TRUE に設定すると、次のように動作します。 • 指定されたアクセスコントロールリスト (ACL) がパブリックの場合、PUT Bucket acl 呼び出しと PUT Object acl 呼び出しは失敗します。 • リクエストにパブリック ACL が含まれていると、PUT Object 呼び出しは失敗します。 • この設定がアカウントに適用されている場合、リクエストにパブリック ACL が含まれていると PUT Bucket の呼び出しは失敗します。 この設定が TRUE に設定されている場合、(REST API、AWS CLI、または AWS SDK のいずれを介して行われたかにかかわらず) 指定された操作は失敗します。ただし、バケットとオブジェクトの既存のポリシーと ACL は変更されません。この設定により、パブリックアクセスから保護しながら、バケットとオブジェクトの既存のポリシーと ACL を監査、絞り込み、またはその他の方法で変更することができます。 **Note** アクセスポイントには ACL が関連付けられていません。この設定をアクセスポイントに適用すると、基になるバケットへのパススルーとして機能します。この設定がアクセスポイントで有効になっている場合、このアクセスポイントを介したリクエストは、基になるバケットでこの設定が実際に有効になっているかどうかに関係

名前	説明
	なく、この設定がバケットで有効になっているかのように動作します。
IgnorePublicAcls	このオプションを TRUE に設定すると、Amazon S3 はバケットとそれに含まれるオブジェクトのすべてのパブリック ACL を無視します。この設定を使用すると、パブリック ACL を含む PUT Object 呼び出しを許可しながら、ACL によって許可されたパブリックアクセスを安全にブロックできます (BlockPublicAcls はパブリック ACL を含む PUT Object 呼び出しを拒否します)。この設定を有効にしても、既存の ACL の永続性には影響せず、新しいパブリック ACL の設定も妨げられません。 **Note** アクセスポイントには ACL が関連付けられていません。この設定をアクセスポイントに適用すると、基になるバケットへのパススルーとして機能します。この設定がアクセスポイントで有効になっている場合、このアクセスポイントを介したリクエストは、基になるバケットでこの設定が実際に有効になっているかどうかに関係なく、この設定がバケットで有効になっているかのように動作します。

名前	説明
BlockPublicPolicy	このオプションをバケットに対し TRUE に設定すると、指定されたバケットポリシーでパブリックアクセスが許可されている場合、Amazon S3 は PUT Bucket ポリシーへの呼び出しを拒否します。バケットに対してこのオプションを TRUE に設定した場合も、Amazon S3 は、指定されたバケットポリシーがパブリックアクセスを許可していれば、バケットのすべての同一アカウントアクセスポイントで PUT アクセスポイントポリシーの呼び出しを拒否します。 アクセスポイントに対してこのオプションを TRUE に設定すると、Amazon S3 は、指定されたポリシー (アクセスポイントまたは基になるバケットのいずれかのポリシー) がパブリックアクセスを許可している場合、アクセスポイントを経由した PUT アクセスポイントポリシーと PUT Bucket ポリシーの呼び出しを拒否します。 この設定を使用することにより、バケットやバケット内のオブジェクトをパブリックに共有することを許可することなく、アクセスポイントとバケットポリシーを管理することをユーザーに許可できます。この設定を有効にしても、既存のアクセスポイントやバケットポリシーには影響しません。 **Important** この設定を効果的に使用するため、アカウントレベルで適用することを推奨します。バケットポリシーでは、ユーザーがバケットのブロックパブリックアクセス設定を変更できます。そのため、バケットポリシーを変更する権限を持つユーザーは、バケットのブロックパブリックアクセス設定を無効にすることを許可するポリシーを挿入できます。この設定が特定のバケットではなくアカウント全体で有効になっている場合、ユーザーがこの設定を無効にするようにバケットポリシーを変更しても、Amazon S3 はパブリックポリシーをブロックします。

名前	説明
RestrictPublicBuckets	このオプションを TRUE に設定すると、パブリックポリシーを持つアクセスポイントやバケットへのアクセスは、バケット所有者のアカウントおよびアクセスポイント所有者のアカウント内の AWS のサービスプリンシパルと承認されたユーザーのみに制限されます。この設定は、アカウント内のユーザーにアクセスポイントやバケットの管理を許可しながら、アクセスポイントやバケットへのすべてのクロスアカウントアクセス (AWS のサービスプリンシパルによるアクセスを除く) をブロックします。 この設定を有効にしても、既存のアクセスポイントポリシーやバケットポリシーには影響しません。ただし、Amazon S3 は、特定のアカウントへのパブリックではない委任を含むパブリックアクセスポイントポリシーやパブリックバケットポリシーから派生したパブリックアクセスやクロスアカウントアクセスをブロックします。

Important

- GET Bucket acl および GET Object acl を呼び出すと、指定されたバケットまたはオブジェクトに対して有効なアクセス許可が常に返されます。例えば、パブリックアクセスを許可する ACL がバケットにあり、そのバケットにも IgnorePublicAcls 設定が有効になっているとします。この場合、GetBucketAcl は、バケットに関連付けられている実際の ACL ではなく、Amazon S3 が強制的に適用しているアクセス許可を反映した ACL を返します。
- ブロックパブリックアクセス設定は既存のポリシーまたは ACL を変更しません。そのため、ブロックパブリックアクセス設定を削除しても、パブリックポリシーまたは ACL を持つバケットまたはオブジェクトは再びパブリックにアクセス可能になります。

アクセスポイントでのパブリックアクセスブロックオペレーションの実行

アクセスポイントに対してブロックパブリックアクセスオペレーションを実行するには、AWS CLI サービス s3control を使用します。

> **Important**
>
> 現在、アクセスポイントの作成後はアクセスポイントのブロックパブリックアクセス設定を変更できないことに注意してください。したがって、アクセスポイントのブロックパブリックアクセス設定を指定する唯一の方法は、アクセスポイントの作成時に設定を含めることです。

「パブリック」の意味

ACL

Amazon S3 は、バケットやオブジェクトの ACL が事前定義済みの `AllUsers` グループまたは `AuthenticatedUsers` グループのメンバーにアクセス許可を付与する場合にパブリックとみなします。事前定義済みのグループの詳細については、Amazon S3 の事前定義済みのグループ を参照してください。

バケットポリシー

バケットポリシーを評価する場合、Amazon S3 はまずポリシーがパブリックであると想定します。その後、ポリシーを評価して非パブリックとしての資格があるかどうかを判断します。非パブリックと見なすには、バケットポリシーで、次のうち 1 つ以上の固定値 (ワイルドカードを含まない値または AWS Identity and Access Managementポリシー変数) にのみアクセスを許可する必要があります。

- AWS プリンシパル、ユーザー、ロール、またはサービスプリンシパル (例: `aws:PrincipalOrgID`)
- `aws:SourceIp` を使用した一連のクラスレスドメイン間ルーティング (CIDR)。CIDR の詳細については、RFC Editor のウェブサイトで RFC 4632 を参照してください。

  > **Note**
  >
  > 非常に広い IP 範囲 (たとえば 0.0.0.0/1) の `aws:SourceIp` 条件キーに基づいてアクセスを許可するバケットポリシーは、「パブリック」と評価されます。これには、IPv4 の場合は /8、IPv6 の場合は /32 よりも広い値が含まれます (RFC1918 のプライベート範囲を除く)。パブリックアクセスをブロックすると、これらの「パブリック」ポリシーが拒否され、これらの「パブリック」ポリシーを既に使用しているバケットへのクロスアカウントアクセスが防止されます。

- `aws:SourceArn`

- aws:SourceVpc
- aws:SourceVpce
- aws:SourceOwner
- aws:SourceAccount
- s3:x-amz-server-side-encryption-aws-kms-key-id
- aws:userid、「AROLEID:*」パターンの外側
- s3:DataAccessPointArn

> **Note**
>
> この値をバケットポリシーで使用すると、アカウント ID が固定されている限り、ポリシーをパブリックにすることなく、アクセスポイント名にワイルドカードを含めることができます。例えば、arn:aws:s3:us-west-2:123456789012:accesspoint/* へのアクセスを許可すると、バケットポリシーをパブリックにすることなく、リージョン 123456789012 のアカウント us-west-2 に関連付けられているすべてのアクセスポイントへのアクセスを許可できます。この動作は、アクセスポイントポリシーでは異なることに注意してください。詳細については、「アクセスポイント」を参照してください。

- s3:DataAccessPointAccount

バケットポリシーの詳細については、Amazon S3 のバケットポリシー を参照してください。

Example : パブリックバケットポリシー

これらのルールでは、次のポリシー例はパブリックと見なされます。

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow"
 }
```

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
```

```
  "Condition": { "StringLike": {"aws:SourceVpc": "vpc-*"}}
 }
```

これらのポリシーは、固定値を使用して、前述のいずれかの条件キーを含めることによって非パブリックにすることができます。例えば、上記の最後のポリシーは、次のように aws:SourceVpc を固定値に設定することで非パブリックにすることができます。

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": {"StringEquals": {"aws:SourceVpc": "vpc-91237329"}}
 }
```

Amazon S3 がパブリックと非パブリックの両方のアクセス許可を含むバケットポリシーを評価する方法

この例では、Amazon S3 がパブリックと非パブリックの両方のアクセス許可を含むバケットポリシーを評価する方法を示します。

バケットに一連の固定プリンシパルへのアクセスを許可するポリシーがあるとします。前述のルールの下では、このポリシーはパブリックではありません。したがって、RestrictPublicBuckets 設定を有効にしても、ポリシーは記述どおりに有効になります。これは、RestrictPublicBuckets はパブリックポリシーを持つバケットにのみ適用されるためです。ただし、パブリックステートメントをポリシーに追加すると、RestrictPublicBuckets はバケットに有効になります。これにより、AWS サービスプリンシパルとバケット所有者のアカウントの認証されたユーザーのみがバケットにアクセスできるようになります。

例えば、「アカウント – 1」が所有するバケットに次の内容を含むポリシーがあるとします。

1. AWS CloudTrail (AWS サービスプリンシパル) へのアクセスを許可するステートメント
2. 「アカウント – 2」アカウントへのアクセスを許可するステートメント
3. Condition 制限なしで "Principal": "*" などを指定して、パブリックへのアクセスを許可するステートメント

このポリシーは、3 番目のステートメントのためにパブリックとしての資格があります。このポリシーがあって RestrictPublicBuckets を有効にすると、Amazon S3 は CloudTrail によるアクセスのみを許可します。ステートメント 2 がパブリックでなくても、Amazon S3 は「アカウント –

2」によるアクセスを無効にします。これは、ステートメント 3 がポリシー全体をパブリックにレンダリングするため、RestrictPublicBuckets が適用されるためです。その結果、ポリシーで特定のアカウントのアカウント – 2」にアクセスが委任されていても、Amazon S3 はクロスアカウントアクセスを無効にします。ただし、ポリシーからステートメント 3 を削除した場合、そのポリシーはパブリックとして認められず、RestrictPublicBuckets は適用されなくなります。したがって、RestrictPublicBuckets を有効のままにしていても、「アカウント – 2」はバケットへのアクセスを再取得できます。

アクセスポイント

Amazon S3 がパブリックアクセスブロック設定を評価する方法は、バケットとアクセスポイントで少しだけ異なります。アクセスポイントポリシーがパブリックであることを確認するために Amazon S3 が適用するルールは、通常、アクセスポイントでもバケットと同じですが、次の場合を除きます。

- VPC ネットワークオリジンを持つアクセスポイントは、アクセスポイントポリシーの内容に関係なく、常に非パブリックと見なされます。
- s3:DataAccessPointArn を使用して一連のアクセスポイントへのアクセスを許可するアクセスポイントポリシーは、パブリックと見なされます。この動作は、バケットポリシーとは異なることに注意してください。例えば、s3:DataAccessPointArn と一致する arn:aws:s3:us-west-2:123456789012:accesspoint/* の値へのアクセスを許可するバケットポリシーは、パブリックと見なされません。ただし、アクセスポイントポリシーにおける同じステートメントは、アクセスポイントをパブリックと見なします。

IAM Access Analyzer for S3 を使用したパブリックバケットの確認

IAM Access Analyzer for S3 を使用して、パブリックアクセスを許可するバケットの ACL、バケットポリシー、またはアクセスポイントポリシーを持つバケットを確認できます。IAM Access Analyzer for S3 は、インターネットの任意のユーザーや他の AWS アカウント (組織外の AWS アカウント を含む) にアクセスを許可するように設定されているバケットに関して警告します。パブリックバケットまたは共有バケットごとに、パブリックアクセスや共有アクセスのソースとレベルを報告する結果が送信されます。

IAM Access Analyzer for S3 では、バケットへのすべてのパブリックアクセスをワンクリックでブロックすることができます。また、バケットレベルのアクセス許可の設定を参照して、きめ細かいアクセスレベルを設定することもできます。パブリックアクセスまたは共有アクセスを必要とする特定の検証済みユースケースについては、バケットの調査結果をアーカイブすることで、バケットをパブリックまたは共有とすることを確定して記録できます。

まれに、Amazon S3 パブリックアクセスブロックの評価でパブリックと報告されたバケットが IAM Access Analyzer for S3 では何も報告されない場合があります。これは、Amazon S3 パブリックアクセスブロックでは、ポリシーの現在のアクションだけでなく、後で追加される可能性があるアクションについても、バケットがパブリックになるかどうかが確認されるためです。一方、IAM Access Analyzer for S3 は、アクセスの状態の評価で Amazon S3 サービスに指定されている現在のアクションのみを分析します。

IAM Access Analyzer for S3 の詳細については、IAM Access Analyzer for S3 を使用したバケットアクセスの確認 を参照してください。

アクセス許可

Amazon S3 のパブリックアクセスブロック機能を使用するには、以下のアクセス許可が必要です。

操作	必要なアクセス許可
GET バケットのポリシーステータス	`s3:GetBucketPolicyStatus`
GET bucket のブロックパブリックアクセス設定	`s3:GetBucketPublicAccessBlock`
PUT bucket のブロックパブリックアクセス設定	`s3:PutBucketPublicAccessBlock`
DELETE bucket のブロックパブリックアクセス設定	`s3:PutBucketPublicAccessBlock`
GET アカウントのブロックパブリックアクセス設定	`s3:GetAccountPublicAccessBlock`
PUT アカウントのブロックパブリックアクセス設定	`s3:PutAccountPublicAccessBlock`
DELETE アカウントのブロックパブリックアクセス設定	`s3:PutAccountPublicAccessBlock`
PUT アクセスポイントのブロックパブリックアクセス設定	`s3:CreateAccessPoint`

> **Note**
>
> DELETE 操作には、PUT オペレーションと同じアクセス許可が必要です。DELETE 操作に対する個別のアクセス許可はありません。

パブリックアクセスブロックの設定

AWS アカウント と Amazon S3 バケットのパブリックアクセスブロック設定の詳細については、以下のトピックを参照してください。

- アカウントのパブリックアクセスブロック設定の構成
- S3 バケットへのパブリックアクセスブロック設定の構成

アカウントのパブリックアクセスブロック設定の構成

Amazon S3 のパブリックアクセスブロックは、Amazon S3 のリソースへのパブリックアクセスの管理に役立つ、アクセスポイント、バケット、アカウントの設定を提供します。デフォルトでは、新しいバケット、アクセスポイント、およびオブジェクトはパブリックアクセスを許可しません。

詳細については、「Amazon S3 ストレージへのパブリックアクセスのブロック」を参照してください。

> **Note**
>
> アカウントレベルの設定は、個々のオブジェクトの設定よりも優先されます。パブリックアクセスをブロックするようにアカウントを設定すると、アカウント内の個々のオブジェクトに対して行われたパブリックアクセスの設定が上書きされます。

S3 コンソール、AWS CLI、AWS SDK、および REST API を使用して、アカウント内のすべてのバケットに対して、パブリックアクセスブロック設定を指定できます。詳細については、関連するセクションを参照してください。

バケットのパブリックアクセスブロック設定を構成するには、「S3 バケットへのパブリックアクセスブロック設定の構成」を参照してください。アクセスポイントに関する情報については、「アクセスポイントでのパブリックアクセスブロックオペレーションの実行」を参照してください。

S3 コンソールの使用

Amazon S3 パブリックアクセスブロックを使用して、S3 バケット内のデータへのパブリックアクセスを許可しないようにアプリケーションを設定します。このセクションでは、AWS アカウントのすべての S3 バケットに対するブロックパブリックアクセス設定の編集方法について説明します。ブロックパブリックアクセスの詳細については、Amazon S3 ストレージへのパブリックアクセスのブロック を参照してください。

AWS アカウントのすべての S3 バケットのブロックパブリックアクセス設定を編集するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Block Public Access settings for this account (このアカウントのブロックパブリックアクセスの設定)] を選択します。
3. [Edit] (編集) を選択して、AWS アカウント のすべてのバケットに対するブロックパブリックアクセス設定を変更します。
4. 変更する設定を選択して、[Save changes (変更の保存)] を選択します。
5. 確認を求められたら、「**`confirm`**」と入力します。次に、[確認] を選択して変更を保存します。

AWS CLI の使用

AWS CLI を介して Amazon S3 のパブリックアクセスブロックを使用することができます。AWS CLI のセットアップおよび使用の詳細については、AWS Command Line Interface とは? を参照してください。

アカウント

アカウントでブロックパブリックアクセス操作を実行するには、AWS CLI サービス `s3control` を使用します。このサービスを使用するアカウントレベルの操作は次のとおりです。

- PUT PublicAccessBlock (アカウントターゲット)
- GET PublicAccessBlock (アカウントターゲット)
- DELETE PublicAccessBlock (アカウントターゲット)

追加情報と例については、AWS CLI リファレンスの put-public-access-block を参照してください。

AWS SDK の使用

Java

以下の例は、AWS SDK for Java で Amazon S3 のパブリックアクセスブロックを使用して、Amazon S3 アカウントにパブリックアクセスブロック設定を配置する方法を示しています。

```
AWSS3ControlClientBuilder controlClientBuilder =
 AWSS3ControlClientBuilder.standard();
controlClientBuilder.setRegion(<region>);
controlClientBuilder.setCredentials(<credentials>);

AWSS3Control client = controlClientBuilder.build();
client.putPublicAccessBlock(new PutPublicAccessBlockRequest()
  .withAccountId(<account-id>)
  .withPublicAccessBlockConfiguration(new PublicAccessBlockConfiguration()
    .withIgnorePublicAcls(<value>)
    .withBlockPublicAcls(<value>)
    .withBlockPublicPolicy(<value>)
    .withRestrictPublicBuckets(<value>)));
```

> **Important**
>
> この例は、AWSS3Control クライアントクラスを使用するアカウントレベルの操作にのみ関係します。バケットレベルの操作については、前述の例を参照してください。

Other SDKs

他の AWS SDK の使用の詳細については、AWS SDK を使用した Amazon S3 での開発 を参照してください。

REST API の使用

REST API での Amazon S3 のパブリックアクセスブロックの使用については、Amazon Simple Storage Service API リファレンスの以下のトピックを参照してください。

- アカウントレベルの操作

- PUT PublicAccessBlock
- GET PublicAccessBlock
- DELETE PublicAccessBlock

S3 バケットへのパブリックアクセスブロック設定の構成

Amazon S3 のパブリックアクセスブロックは、Amazon S3 のリソースへのパブリックアクセスの管理に役立つ、アクセスポイント、バケット、アカウントの設定を提供します。デフォルトでは、新しいバケット、アクセスポイント、およびオブジェクトはパブリックアクセスを許可しません。

詳細については、「Amazon S3 ストレージへのパブリックアクセスのブロック」を参照してください。

S3 コンソール、AWS CLI、AWS SDK、REST API を使用して、1 つ以上のバケットへのパブリックアクセスを許可できます。また、既にパブリックになっているバケットへのパブリックアクセスをブロックすることもできます。詳細については、関連するセクションを参照してください。

アカウント内のすべてのバケットに対してパブリックアクセスをブロックする設定を構成するには、「アカウントのパブリックアクセスブロック設定の構成」を参照してください。アクセスポイントのパブリックアクセスブロック設定については、「アクセスポイントでのパブリックアクセスブロックオペレーションの実行」を参照してください。

S3 コンソールの使用

Amazon S3 パブリックアクセスブロックを使用して、S3 バケット内のデータへのパブリックアクセスを許可しないようにアプリケーションを設定します。このセクションでは、1 つ以上の S3 バケットに対するブロックパブリックアクセス設定の編集方法について説明します。AWS CLI、AWS SDK、および Amazon S3 REST API を使用したパブリックアクセスのブロックについて詳細は、Amazon S3 ストレージへのパブリックアクセスのブロック を参照してください。

[Buckets] (バケット) リストでバケットがパブリックにアクセス可能であるかどうかを確認できます。[Access] (アクセス) 列で、Amazon S3 はバケットのアクセス権限を次のようにラベル付けします。

- パブリック – すべてのユーザーにオブジェクトのリスト権限、オブジェクトの書き込み権限、読み取りと書き込み権限のうちの 1 つまたは複数が与えられています。
- オブジェクトは公開可能 – バケットはパブリックではありませんが、適切なアクセス許可を持つユーザーは、オブジェクトへのパブリックアクセスを許可できます。

- バケットとオブジェクトは非公開 – バケットとオブジェクトにはパブリックアクセス許可がありません。
- このアカウントの承認済ユーザーのみ – パブリックアクセスを許可するポリシーがあるため、アクセスはこのアカウントおよび AWS のサービスプリンシパルの IAM ユーザーおよびロールに限定されます。

アクセスタイプでバケット検索をフィルタリングすることもできます。[バケット検索] バーの横にあるドロップダウンリストからアクセスタイプを選択します。

バケットとそのパブリックアクセス設定を一覧表示したときに `Error` が表示された場合は、必要なアクセス許可がない可能性があります。以下のアクセス許可がユーザーポリシーまたはロールポリシーに追加されていることを確認します。

```
s3:GetAccountPublicAccessBlock
s3:GetBucketPublicAccessBlock
s3:GetBucketPolicyStatus
s3:GetBucketLocation
s3:GetBucketAcl
s3:ListAccessPoints
s3:ListAllMyBuckets
```

まれに、AWS リージョンの停止が原因で、リクエストが失敗することもあります。

単一の S3 バケットをターゲットに Amazon S3 パブリックアクセスブロック設定を編集するには

単一の S3 バケットのパブリックアクセス設定を変更する必要がある場合は、以下の手順に従ってください。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット名] リストで、目的のバケットの名前を選択します。
3. [Permissions] を選択します。
4. [Edit] (編集) をクリックして、バケットのパブリックアクセス設定を変更します。4 つの Amazon S3 パブリックアクセスブロック設定の詳細については、パブリックアクセスブロック設定 を参照してください。
5. 変更する設定を選択して、[保存] を選択します。
6. 確認を求められたら、「**`confirm`**」と入力します。次に、[確認] を選択して変更を保存します。

バケットを作成するとき、Amazon S3 パブリックアクセスブロックの設定を変更できます。詳細については、「バケットの作成」を参照してください。

AWS CLI の使用

バケットのパブリックアクセスをブロックしたり、パブリックアクセスのブロックを削除するには、AWS CLI サービス `s3api` を使用します。このサービスを使用するバケットレベルの操作は次のとおりです。

- PUT PublicAccessBlock (バケットターゲット)
- GET PublicAccessBlock (バケットターゲット)
- DELETE PublicAccessBlock (バケットターゲット)
- GET BucketPolicyStatus

詳細と例については、AWS CLI リファレンスの put-public-access-block を参照してください。

AWS SDK の使用

Java

```
AmazonS3 client = AmazonS3ClientBuilder.standard()
   .withCredentials(<credentials>)
   .build();

client.setPublicAccessBlock(new SetPublicAccessBlockRequest()
  .withBucketName(<bucket-name>)
  .withPublicAccessBlockConfiguration(new PublicAccessBlockConfiguration()
    .withBlockPublicAcls(<value>)
    .withIgnorePublicAcls(<value>)
    .withBlockPublicPolicy(<value>)
    .withRestrictPublicBuckets(<value>)));
```

Important

この例は、`AmazonS3` クライアントクラスを使用するバケットレベルの操作にのみ関係します。アカウントレベルの操作については、次の例を参照してください。

Other SDKs

他の AWS SDK の使用の詳細については、AWS SDK を使用した Amazon S3 での開発 を参照してください。

REST API の使用

REST API での Amazon S3 のパブリックアクセスブロックの使用については、Amazon Simple Storage Service API リファレンスの以下のトピックを参照してください。

- バケットレベルの操作
  - PUT PublicAccessBlock
  - GET PublicAccessBlock
  - DELETE PublicAccessBlock
  - GET BucketPolicyStatus

IAM Access Analyzer for S3 を使用したバケットアクセスの確認

IAM Access Analyzer for S3 は、インターネットの任意のユーザーや他の AWS アカウント (組織外の AWS アカウントを含む) にアクセスを許可するように設定されている S3 バケットに関して警告します。パブリックバケットまたは共有バケットごとに、パブリックアクセスや共有アクセスのソースとレベルを示す結果が送信されます。例えば、IAM Access Analyzer for S3 は、バケットのアクセスコントロールリスト (ACL)、バケットポリシー、マルチリージョンアクセスポイントポリシー、またはアクセスポイントポリシーを通じて、バケットに読み取りまたは書き込みのアクセス許可が提供されていることを示す場合があります。この情報を用いて、迅速で正確な是正処置を講じ、バケットへのアクセスを意図したとおりに復元できます。

IAM Access Analyzer for S3 でリスクのあるバケットを確認した場合、ワンクリックでバケットへのすべてのパブリックアクセスをブロックできます。特定のユースケースをサポートするためにパブリックアクセスが必要な場合を除き、バケットへのすべてのアクセスをブロックすることをお勧めします。すべてのパブリックアクセスをブロックする前に、アプリケーションがパブリックアクセスなしで正常に動作することを確認してください。詳細については、「Amazon S3 ストレージへのパブリックアクセスのブロック」を参照してください。

また、バケットレベルのアクセス許可の設定を参照して、きめ細かいアクセスレベルを設定することもできます。静的なウェブサイトホスティング、パブリックダウンロード、クロスアカウント共有など、パブリックアクセスを必要とする特定の検証済みユースケースについては、バケットの結果を

アーカイブすることで、バケットをパブリックまたは共有とすることを確定して記録できます。これらのバケット設定はいつでも再確認および変更できます。結果は、監査目的で CSV レポートとしてダウンロードすることもできます。

IAM Access Analyzer for S3 は、Amazon S3 コンソールで追加料金なしで使用できます。IAM Access Analyzer for S3 は、AWS Identity and Access Management (IAM) IAM Access Analyzer を利用しています。Amazon S3 コンソールで IAM Access Analyzer for S3 を使用するには、IAM コンソールにアクセスして、リージョンごとに IAM Access Analyzer を有効にする必要があります。

IAM Access Analyzer の詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer とは?」を参照してください。IAM Access Analyzer for S3 の詳細については、次のセクションを参照してください。

Important

- IAM Access Analyzer for S3 には、アカウントレベルのアナライザーが必要です。IAM Access Analyzer for S3 を使用するには、IAM Access Analyzer にアクセスして、信頼ゾーンとしてアカウントを持つアナライザーを作成する必要があります。詳細については、IAM ユーザーガイド の「IAM Access Analyzer の有効化」を参照してください。
- IAM Access Analyzer for S3 は、クロスアカウントアクセスポイントにアタッチされているアクセスポイントポリシーを分析しません。この動作は、アクセスポイントとそのポリシーが信頼ゾーン、つまりアカウントの外にあるために発生します。クロスアカウントアクセスポイントへのアクセスを委任するバケットは、バケットまたはアカウントにパブリックアクセスをブロックする `RestrictPublicBuckets` 設定を適用していない場合、[Buckets with public access] (パブリックアクセスのあるバケット) に一覧表示されます。パブリックアクセスをブロックする設定 `RestrictPublicBuckets` を適用すると、そのバケットは、[サードパーティーの AWS アカウント を含む他の AWS アカウント からのアクセスがあるバケット] に表示されます。
- バケットポリシーまたはバケット ACL を追加または変更すると、IAM Access Analyzer は 30 分以内に変更に基づいて結果を生成および更新します。アカウントレベルのブロックパブリックアクセス設定に関連する結果は、設定を変更してから最大 6 時間まで生成または更新されない場合があります。マルチリージョンアクセスポイントに関連する調査結果は、マルチリージョンアクセスポイントの作成、削除、またはポリシーの変更後、最大 6 時間は生成または更新できない場合があります。

トピック

IAM Access Analyzer for S3 はどのような情報を提供しますか？

IAM Access Analyzer for S3 は、AWS アカウント 外からアクセスできるバケットに関する結果を提供します。[Buckets with public access (パブリックアクセスを許可するバケット)] に表示されるバケットには、インターネットの任意のユーザーがアクセスできます。IAM Access Analyzer for S3 がパブリックバケットを特定すると、リージョン内のパブリックバケット数を示す警告もページの上部に表示されます。第三者を含む、他の AWS アカウント からアクセスのあるバケット — サード・パーティーを含む AWS アカウントでリスト化したバケットは、組織外のアカウントを含む他の AWS アカウント という条件付きで共有されます。

IAM Access Analyzer for S3 は、バケットごとに次の情報を提供します。

- バケット名
- Access Analyzer によって検出 – IAM Access Analyzer for S3 がパブリックまたは共有バケットアクセスを検出したとき。
- 共有方法 – バケットの共有方法 — バケットポリシー、バケット ACL、マルチリージョンアクセスポイントポリシー、またはアクセスポイントポリシーを通じて。マルチリージョンアクセスポイントおよびクロスアカウントアクセスポイントは、アクセスポイントの下に反映されます。バケットは、ポリシーと ACL の両方を通じて共有できます。バケットアクセスのソースを検索して確認する場合は、この列の情報をまず使用して、迅速で正確な是正措置を実行できます。
- ステータス - バケット結果のステータス。IAM Access Analyzer for S3 には、すべてのパブリックバケットと共有バケットの検出結果が表示されます。
  - アクティブ – 結果は確認されていません。
  - アーカイブ済み – 結果は意図したとおりにレビューおよび確認されています。
  - すべて - パブリックバケット、または他の AWS アカウント (組織外の AWS アカウント を含む) との共有バケットに関するすべての結果。

- アクセスレベル – バケットに付与されているアクセス許可:
  - リスト – リソースを一覧表示します。
  - 読み取り – リソースの内容と属性を読み取ります (ただし、編集しません)。
  - 書き込み – リソースを作成、削除、または変更します。
  - アクセス許可 – リソースに対するアクセス許可を付与または変更します。
  - タグ付け – リソースに関連付けられているタグを更新します。

IAM Access Analyzer for S3 の有効化

IAM Access Analyzer for S3 を使用するには、前提条件に関する以下のステップを完了する必要があります。

1. 必要なアクセス権限を付与します。

   詳細については、IAM ユーザーガイドの「IAM Access Analyzer の使用に必要なアクセス許可」を参照してください。

2. IAM にアクセスして、IAM Access Analyzer を使用するリージョンごとにアカウントレベルのアナライザーを作成します。

   IAM Access Analyzer for S3 には、アカウントレベルのアナライザーが必要です。IAM Access Analyzer for S3 を使用するには、信頼ゾーンとしてアカウントを持つアナライザーを作成する必要があります。詳細については、IAM ユーザーガイド の「IAM Access Analyzer の有効化」を参照してください。

すべてのパブリックアクセスのブロック

バケットへのすべてのアクセスをワンクリックでブロックする場合は、IAM Access Analyzer for S3 の [Block all public access] (すべてのパブリックアクセスをブロック) ボタンを使用できます。バケットへのすべてのパブリックアクセスをブロックすると、一切のパブリックアクセスが許可されません。特定の検証済みユースケースをサポートするためにパブリックアクセスが必要な場合を除き、バケットへのすべてのパブリックアクセスをブロックすることをお勧めします。すべてのパブリックアクセスをブロックする前に、アプリケーションがパブリックアクセスなしで正常に動作することを確認してください。

バケットへのすべてのパブリックアクセスをブロックしたくない場合は、Amazon S3 コンソールでブロックパブリックアクセス設定を編集して、バケットへの詳細なアクセスレベルを設定できます。

詳細については、「Amazon S3 ストレージへのパブリックアクセスのブロック」を参照してください。

まれに、Amazon S3 パブリックアクセスブロックの評価でパブリックと報告されたバケットが IAM Access Analyzer for S3 では何も報告されない場合があります。これは、Amazon S3 パブリックアクセスブロックでは、ポリシーの現在のアクションだけでなく、後で追加される可能性があるアクションについても、バケットがパブリックになるかどうかが確認されるためです。一方、IAM Access Analyzer for S3 は、アクセスの状態の評価で Amazon S3 サービスに指定されている現在のアクションのみを分析します。

IAM Access Analyzer for S3 を使用してバケットへのすべてのパブリックアクセスをブロックするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインの [ダッシュボード] で、[Access analyzer for S3 (S3 のアクセスアナライザー)] を選択します。
3. IAM Access Analyzer for S3 で、バケットを選択します。
4. [Block all public access (すべてのパブリックアクセスをブロック)] を選択します。
5. バケットへのすべてのパブリックアクセスをブロックすることを確定するには、[Block all public access (bucket settings) (すべてのパブリックアクセスをブロック (バケット設定))] に「**`confirm`**」と入力します。

   Amazon S3 は、バケットへのすべてのパブリックアクセスをブロックします。バケットの検出結果のステータスが [解決済み] に更新され、バケットが IAM Access Analyzer for S3 のリストから消えます。解決済みのバケットを確認する場合は、IAM コンソールで IAM Access Analyzer を開きます。

バケットアクセスの確認と変更

パブリックアカウントや他の AWS アカウント アカウント (組織外のアカウントを含む) にアクセスを許可しない場合は、バケット ACL、バケットポリシー、マルチリージョンアクセスポイントポリシー、またはアクセスポイントポリシーを変更してバケットへのアクセスを削除できます。[Shared Through] 列には、バケットアクセスのすべてのソース (バケットポリシー、バケット ACL、アクセスポイントポリシー) が表示されます。マルチリージョンアクセスポイントおよびクロスアカウントアクセスポイントは、アクセスポイントの下に反映されます。

バケットポリシー、バケット ACL、マルチリージョンアクセスポイント、またはアクセスポイントポリシーを確認および変更するには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. ナビゲーションペインで、[S3 用 Access Analyzer] を選択します。
3. パブリックアクセスまたは共有アクセスが、バケットポリシー、バケット ACL、マルチリージョンアクセスポイントポリシー、またはアクセスポイントポリシーを通じて許可されているかどうかを確認するには、[Shared through] 列を調べます。
4. [バケット] で、バケットポリシー、バケット ACL、マルチリージョンアクセスポイントポリシー、またはアクセスポイントポリシーを変更または確認するバケットの名前を選択します。
5. バケット ACL を変更または表示するには

   a. [Permissions] を選択します。

   b. [Access Control List] を選択します。

   c. バケット ACL を確認し、必要に応じて変更します。

      詳細については、「ACL の設定」を参照してください。
6. バケットポリシーを変更または確認するには

   a. [Permissions] を選択します。

   b. [バケットポリシー] を選択します。

   c. 必要に応じて、バケットポリシーを確認または変更します。

      詳細については、「Amazon S3 コンソールを使用したバケットポリシーの追加」を参照してください。
7. マルチリージョンアクセスポイントポリシーを確認または変更するには

   a. [マルチリージョンアクセスポイント] を選択します。

   b. マルチリージョンアクセスポイント名を選択します。

   c. 必要に応じて、マルチリージョンアクセスポイントポリシーを確認または変更します。

      詳細については、「許可」を参照してください。
8. アクセスポイントポリシーを確認または変更するには

   a. [access points] (アクセスポイント) を選択します。

   b. アクセスポイント名を選択します。

c. 必要に応じてアクセスを確認または変更します。

詳細については、「Amazon S3 コンソールでの Amazon S3 アクセスポイントの使用」を参照してください。

バケット ACL、バケットポリシー、アクセスポイントポリシーを編集または削除してパブリックアクセスまたは共有アクセスを削除すると、バケットの結果のステータスが解決済みに更新されます。解決済みのバケットの検出結果は IAM Access Analyzer for S3 リストから消えますが、IAM Access Analyzer で表示できます。

バケットの結果のアーカイブ

バケットが、特定のユースケース (静的ウェブサイト、パブリックダウンロード、クロスアカウント共有など) をサポートするために、パブリックアカウントや他の AWS アカウント (組織外のアカウントを含む) へのアクセスを許可する場合は、バケットの結果をアーカイブできます。バケットの結果をアーカイブする場合、バケットをパブリックまたは共有にすることを確定して記録します。アーカイブされたバケットの検出結果は IAM Access Analyzer for S3 リストに残るため、どれがパブリックバケットまたは共有バケットであるかを常に把握できます。

IAM Access Analyzer for S3 でバケットの検出結果をアーカイブするには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. ナビゲーションペインで、[S3 用 Access Analyzer] を選択します。
3. IAM Access Analyzer for S3 で、アクティブなバケットを選択します。
4. このバケットにパブリックアカウントや他の AWS アカウント (組織外のアカウントを含む) からアクセスすることを確定するには、[Archive] (アーカイブ) を選択します。
5. 「**`confirm`**」と入力し、[アーカイブ] を選択します。

アーカイブされたバケットの結果の有効化

結果のアーカイブ後は、いつでも結果に再アクセスし、ステータスをアクティブに戻すことができます。この場合、バケットには別のレビューが必要になります。

アーカイブされたバケットの検出結果を IAM Access Analyzer for S3 でアクティブにするには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. ナビゲーションペインで、[S3 用 Access Analyzer] を選択します。

3. アーカイブされたバケットの結果を選択します。
4. [Mark as active (アクティブとしてマーク)] を選択します。

結果の詳細の表示

バケットの詳細情報を表示する必要がある場合は、IAM コンソールの IAM Access Analyzer でバケットの結果の詳細を開くことができます。

IAM Access Analyzer for S3 で検出結果の詳細を表示するには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. ナビゲーションペインで、[S3 用 Access Analyzer] を選択します。
3. IAM Access Analyzer for S3 で、バケットを選択します。
4. [View details] (詳細を表示する) をクリックします。

   結果の詳細が IAM コンソールの IAM Access Analyzer で開きます。

IAM Access Analyzer for S3 レポートのダウンロード

バケットの結果を CSV レポートとしてダウンロードし、監査目的に使用できます。レポートには、Amazon S3 コンソールの IAM Access Analyzer for S3 に表示される情報と同じ情報が含まれます。

レポートをダウンロードするには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 左側のナビゲーションペインで、[Access analyzer for S3 (S3 のアクセスアナライザー)] を選択します。
3. [リージョンフィルタ] で、[リージョン] を選択します。

   IAM Access Analyzer for S3 が更新され、選択したリージョンのバケットが表示されます。
4. [レポートのダウンロード] を選択します。

   CSV レポートが生成され、コンピュータに保存されます。

バケット所有者条件によるバケット所有者の確認

Amazon S3 バケット所有者条件では、S3 オペレーションで使用するバケットが適切な AWS アカウントに属していることを確認します。

ほとんどの S3 オペレーションでは、特定の S3 バケットに対して読み取りまたは書き込みを行います。これらのオペレーションには、オブジェクトのアップロード、コピー、ダウンロード、バケット設定の取得または変更、オブジェクト設定の取得または変更が含まれます。これらのオペレーションを実行するときは、リクエストにバケット名を含めることによって、使用するバケットを指定します。例えば、S3 からオブジェクトを取得するには、バケットの名前と、そのバケットから取得するオブジェクトキーを指定してリクエストを作成します。

Amazon S3 は、名前に基づいてバケットを識別するため、アプリケーションでリクエストに誤ったバケット名を使用した場合、正しくないバケットに対して、誤ってオペレーションが実行される可能性があります。このような状況で意図しないバケットとのやり取りを回避するために、バケット所有者条件を使用できます。バケット所有者条件を使用すると、ターゲットバケット所有者が正しい AWS アカウント であることを確認して、S3 オペレーションの効果が意図どおりになることをさらに保証できます。

トピック

バケット所有者条件を使用する状況

正しいバケット所有者のアカウント ID がわかっていて、サポートされている S3 オペレーションを実行する場合は、バケット所有者条件を使用することをお勧めします。バケット所有者条件は、すべての S3 オブジェクトオペレーションおよびほとんどの S3 バケットオペレーションで使用できます。バケット所有者条件がサポートされない S3 オペレーションのリストについては、制約と制限を参照してください。

バケット所有者条件を使用するメリットを確認するには、AWS の顧客である Bea に関係する次のシナリオを考察してください。

1. Bea は Amazon S3 を使用するアプリケーションを開発しています。開発中、Bea はテスト専用の AWS アカウントを使用して bea-data-test という名前のバケットを作成し、アプリケーションで bea-data-test にリクエストが実行されるように設定しました。
2. Bea はアプリケーションをデプロイしましたが、本稼働用の AWS アカウントでバケットを使用するようにアプリケーションを再設定するのを忘れました。
3. 本稼働環境では、Bea のアプリケーションが bea-data-test にリクエストを行い、その実行に成功しました。その結果、Bea のテストアカウントのバケットに本稼働用データが書き込まれます。

Bea は、バケット所有者条件を使用することで、このような状況を回避できます。バケット所有者条件を使用すると、Bea は正しいバケット所有者の AWS アカウント ID をリクエストに含めることができます。Amazon S3 は、各リクエストを処理する前に、バケット所有者のアカウント ID をチェックします。実際のバケット所有者が正しいバケット所有者と一致しない場合、リクエストは失敗します。

Bea がバケット所有者条件を使用する場合、前述のシナリオで Bea のアプリケーションが誤ってテストバケットに書き込むことにはなりません。その代わりに、ステップ 3 でアプリケーションが行うリクエストは失敗し、Access Denied エラーメッセージが表示されます。バケット所有者条件を使用することで、Bea は正しくない AWS アカウント のバケットと誤ってやり取りするリスクを排除できます。

バケット所有者の確認

バケット所有者条件を使用するには、正しいバケット所有者を指定するパラメータをリクエストに含めます。ほとんどの S3 オペレーションに含まれるバケットは 1 つだけであり、バケット所有者条件を使用するために必要になるのは、この 1 つのパラメータのみです。CopyObject オペレーションの場合、この最初のパラメータでは宛先バケットの正しい所有者を指定します。ソースバケットの正しい所有者を指定するには、2 番目のパラメータを含めます。

バケット所有者条件パラメータを含むリクエストを行うと、S3 はリクエストを処理する前に、指定されたパラメータに対してバケット所有者のアカウント ID をチェックします。パラメータがバケット所有者のアカウント ID と一致すると、S3 がリクエストを処理します。パラメータがバケット所有者のアカウント ID と一致しない場合、リクエストは失敗し、Access Denied エラーメッセージが表示されます。

バケット所有者条件は、AWS Command Line Interface (AWS CLI)、AWS SDK、および Amazon S3 REST API で使用できます。AWS CLI および Amazon S3 REST API でバケット所有者条件を使用する場合は、次のパラメータ名を使用します。

アクセス方法	コピーオペレーション以外のパラメータ	コピーオペレーションのソースパラメータ	コピーオペレーションの宛先パラメータ
AWS CLI	`--expected-bucket-owner`	`--expected-source-bucket-owner`	`--expected-bucket-owner`
Amazon S3 REST API	`x-amz-expected-bucket-owner` ヘッダー	`x-amz-source-expected-bucket-owner` ヘッダー	`x-amz-expected-bucket-owner` ヘッダー

AWS SDK でバケット所有者条件を使用するために必要なパラメータ名は、言語によって異なります。必要なパラメータを確認するには、使用する言語の SDK ドキュメント を参照してください。SDK ドキュメントには、AWS での構築ツールからアクセスできます。

例

以下の例は、AWS CLI または AWS SDK for Java 2.x を使用してバケット所有者条件を Amazon S3 に実装する方法を示しています。

Example

例: オブジェクトのアップロード

次の例では、バケット所有者条件を使用して S3 バケット *example-s3-bucket1* にオブジェクトをアップロードし、*example-s3-bucket1* の所有者が AWS アカウント 111122223333 であることを確認します。

AWS CLI

```
aws s3api put-object \
                --bucket example-s3-bucket1 --key exampleobject --
body example_file.txt \
                --expected-bucket-owner 111122223333
```

AWS SDK for Java 2.x

```
public void putObjectExample() {
```

```
    S3Client s3Client = S3Client.create();;
    PutObjectRequest request = PutObjectRequest.builder()
            .bucket("example-s3-bucket1")
            .key("exampleobject")
            .expectedBucketOwner("111122223333")
            .build();
    Path path = Paths.get("example_file.txt");
    s3Client.putObject(request, path);
}
```

Example

例: オブジェクトのコピー

次の例では、S3 バケット *example-s3-bucket1* から S3 バケット *example-s3-bucket2* にオブジェクト object1 をコピーします。この例では、バケット所有者条件を使用し、次の表に従って、バケットを所有するアカウントが正しいことを確認します。

バケット	正しい所有者
example-s3-bucket1	111122223333
example-s3-bucket2	444455556666

AWS CLI

```
aws s3api copy-object --copy-source example-s3-bucket1/object1 \
                            --bucket example-s3-bucket2 --key object1copy \
                            --expected-source-bucket-owner 111122223333 --expected-
bucket-owner 444455556666
```

AWS SDK for Java 2.x

```
public void copyObjectExample() {
        S3Client s3Client = S3Client.create();
        CopyObjectRequest request = CopyObjectRequest.builder()
                .copySource("example-s3-bucket1/object1")
                .destinationBucket("example-s3-bucket2")
                .destinationKey("object1copy")
```

```
                .expectedSourceBucketOwner("111122223333")
                .expectedBucketOwner("444455556666")
                .build();
        s3Client.copyObject(request);
    }
```

Example

例: バケットポリシーの取得

次の例では、バケット所有者条件を使用して S3 バケット *example-s3-bucket1* のアクセスポリシーを取得し、*example-s3-bucket1* の所有者が AWS アカウント 111122223333 であることを確認します。

AWS CLI

```
aws s3api get-bucket-policy --bucket example-s3-bucket1 --expected-bucket-
owner 111122223333
```

AWS SDK for Java 2.x

```
public void getBucketPolicyExample() {
    S3Client s3Client = S3Client.create();
    GetBucketPolicyRequest request = GetBucketPolicyRequest.builder()
            .bucket("example-s3-bucket1")
            .expectedBucketOwner("111122223333")
            .build();
    try {
        GetBucketPolicyResponse response = s3Client.getBucketPolicy(request);
    }
    catch (S3Exception e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.
        e.printStackTrace();
    }
}
```

制約と制限

Amazon S3 バケット所有者条件には、次の制約と制限があります。

- バケット所有者条件パラメータの値は、AWS アカウント ID (12 桁の数値) である必要があります。サービスプリンシパルはサポートされていません。
- バケット所有者条件は、CreateBucket、ListBuckets、または AWS S3 コントロールに含まれるオペレーションでは使用できません。Amazon S3 では、これらのオペレーションに対するリクエストに含まれるバケット所有者条件パラメータがすべて無視されます。
- バケット所有者条件は、検証パラメータで指定されたアカウントがバケットを所有しているかどうかのみを検証します。バケット所有者条件で、バケットの設定はチェックされません。また、バケットの設定が特定の条件を満たしていること、または過去の状態と一致することを保証するものではありません。

オブジェクトの所有権の制御とバケットの ACL の無効化。

S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、バケットにアップロードされたオブジェクトの所有権を制御し、アクセスコントロールリスト (ACL) を有効または無効にするのに使用できます。デフォルトでは、オブジェクト所有権は [バケット所有者の強制] により設定され、すべての ACL は無効になっています。ACL が無効になっている場合、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。

Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。そのため、オブジェクトごとに個別にアクセスを制御する必要がある異常な状況を除き、ACL を無効にしておくことをお勧めします。ACL を無効にすると、バケット内のオブジェクトをアップロードしたユーザーに関係なく、ポリシーを使用してバケット内のすべてのオブジェクトへのアクセスをより簡単に制御できます。

オブジェクト所有権には、バケットにアップロードされるオブジェクトの所有権を制御し、ACL を無効または有効化するために使用できる 3 つの設定があります。

ACL は無効です

- バケット所有者強制 (デフォルト) – ACL は無効になり、バケット所有者はバケット内のすべてのオブジェクトを自動的に所有し、完全に制御します。ACL は、S3 バケット内のデータに対するアクセス許可に対して影響を与えません。バケットは、ポリシーを使用してアクセスコントロールを定義します。

ACL の有効化

- 希望するバケット所有者 — バケット所有者は、他のアカウントが bucket-owner-full-control 既定 ACL でバケットに書き込む新しいオブジェクトを所有し、完全にコントロールできます。
- オブジェクトライター — オブジェクトをアップロードする AWS アカウント は、そのオブジェクトを所有し、そのオブジェクトを完全にコントロールし、ACL を介して他のユーザーにそのオブジェクトへのアクセス権を付与できます。

S3 の最近のユースケースの大部分では、[バケット所有者の強制] 設定を適用して ACL を無効なままにし、必要に応じてバケットポリシーを使用して、アカウント外のユーザーとデータを共有することをお勧めします。このアプローチにより、アクセス許可の管理が簡素化されます。新しく作成されたバケットと既存のバケットの両方で ACL を無効にすることができます。新しく作成されたバケットでは、ACL はデフォルトで無効になっています。既存のバケットに既にオブジェクトが含まれている場合、ACL を無効にすると、オブジェクトとバケット ACL はアクセス評価に含まれなくなり、ポリシーに基づいてアクセスが許可または拒否されます。既存のバケットについては、ACL を無効にした後いつでも再度有効化でき、既存のバケットとオブジェクト ACL が復元されます。

ACL を無効にする前に、バケットポリシーを確認して、アカウント外のバケットへのアクセス権を付与するすべての方法をカバーすることを確認することをお勧めします。ACL を無効にすると、バケットは ACL を指定しない PUT リクエスト、またはバケット所有者のフルコントロール ACL (bucket-owner-full-control 既定の ACL またはこの ACL と同等の XML で表される形式など) を持つ PUT リクエストのみを受け入れます。バケット所有者のフルコントロール ACL をサポートする既存のアプリケーションには影響はありません。他の ACL (特定の AWS アカウント へのカスタム許可など) を含む PUT リクエストは失敗し、AccessControlListNotSupported エラーコードを含む 400 エラーを返します。

それとは対照に、[バケット所有者が推奨] で設定されているバケットは、引き続きバケットおよびオブジェクト ACL を受け入れ、遵守します。この設定では、新しいオブジェクトが bucket-owner-full-control 既定 ACL はオブジェクトライターではなく、バケット所有者によって自動的に所有されます。その他のすべての ACL 動作はそのまま残ります。すべての Amazon S3 PUT 操作に bucket-owner-full-control 既定 ACL を含めるように要求するには、この ACL を使用したオブジェクトのアップロードのみを許可するバケットポリシーを追加できます。

どのオブジェクト所有権設定がバケットに適用されているかを確認するには、Amazon S3 ストレージレンズメトリクスを使用できます。S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージ分析機能です。詳細

については、「Using S3 Storage Lens to find Object Ownership settings」(S3 ストレージレンズを使用してオブジェクト所有権の設定を検索する) を参照してください。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

オブジェクトの所有権の設定

この表は、各オブジェクト所有権設定が ACL、オブジェクト、オブジェクト所有権、およびオブジェクトアップロードに与える影響を示しています。

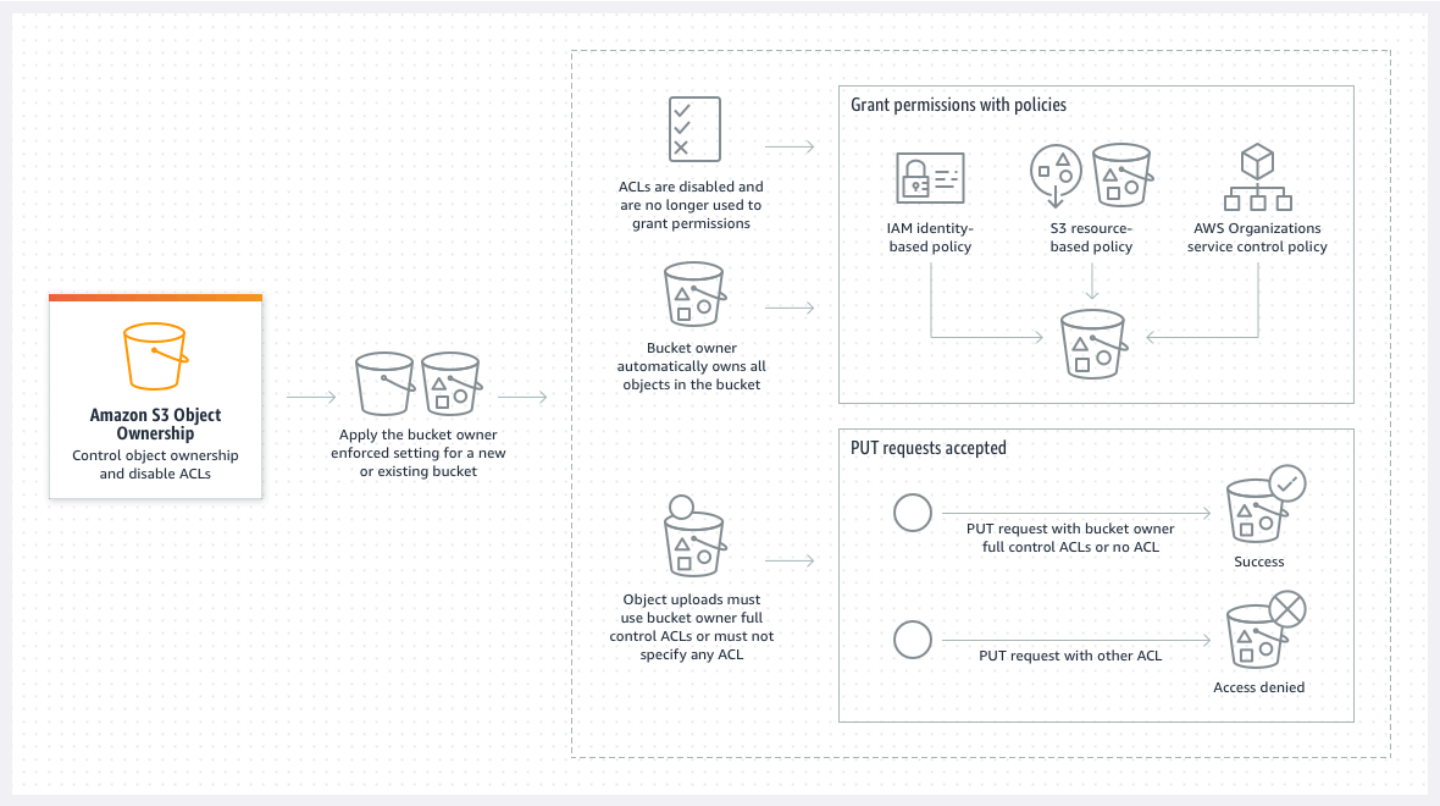
設定	適用先	オブジェクトの所有者への影響	ACL への影響	アップロードを受け付けました
バケット所有者強制 (デフォルト)	既存のオブジェクトと新しいオブジェクト	バケット所有者はすべてのオブジェクトを所有しています。	ACL は無効になり、バケットへのアクセス権限には影響しません。ACL を設定または更新する要求は失敗します。ただし、ACL の読み取り要求はサポートされています。 バケット所有者は完全な所有権と制御を有しています。 オブジェクトライターには完全	バケット所有者の完全制御 ACL を使用したアップロード、または ACL を指定しないアップロード。

設定	適用先	オブジェクトの所有者への影響	ACL への影響	アップロードを受け付けました
			な所有権と制御がなくなりました。	
バケット所有者推奨。	新しいオブジェクト	オブジェクトのアップロードが bucket-owner-full-control 既定 ACL を含む場合、バケット所有者はオブジェクトを所有しています。 他の ACL でアップロードしたオブジェクトは、書き込みアカウントによって所有されます。	ACL は更新でき、許可を付与できます。 オブジェクトのアップロードが bucket-owner-full-control 既定 ACL を含む場合、バケット所有者は完全制御アクセスであり、オブジェクトライターは完全制御アクセスではありません。	すべてアップロードします。
オブジェクトライター	新しいオブジェクト	オブジェクトライターがオブジェクトを所有します。	ACL は更新でき、許可を付与できます。 オブジェクトライターは完全制御アクセスを有します。	すべてアップロードします。

ACL を無効にして導入された変更

オブジェクト所有権に対して [バケット所有者の強制] 設定を適用すると、ACL は無効になり、追加のアクションを実行せずに、バケット内のすべてのオブジェクトを自動的に所有し、完全に制御できます。[バケット所有者の強制] 設定は、新しく作成されたすべてのバケットでデフォルト設定となっています。[バケット所有者の強制] 設定が適用されると、3 つの変更が表示されます。

- すべてのバケット ACL とオブジェクト ACL が無効になり、バケット所有者としてフルアクセスが付与されます。バケットまたはオブジェクトに対して読み取り ACL リクエストを実行すると、バケット所有者にのみフルアクセスが付与されていることがわかります。
- バケット所有者は、バケット内のすべてのオブジェクトを自動的に所有し、完全に制御できます。
- ACL はバケットへのアクセス許可に影響を与えなくなりました。その結果、データのアクセスコントロールは、IAM ポリシー、S3 バケットポリシー、VPC エンドポイントポリシー、組織 SCP などのポリシーに基づいています。



S3 バージョニングを使用する場合、バケット所有者はバケット内のすべてのオブジェクトバージョンを所有し、完全に制御できます。[バケット所有者の強制] 設定を適用しても、オブジェクトの新しいバージョンは追加されません。

新しいオブジェクトをバケットにアップロードできるのは、バケット所有者の完全制御 ACL を使用するか、ACL を指定しない場合のみです。オブジェクトのアップロードは、他の ACL を指定すると失敗します。詳細については、「トラブルシューティング」を参照してください。

次の例では PutObject (AWS Command Line Interface) を使用した AWS CLI オペレーション を含む bucket-owner-full-control 既定 ACL の場合、無効な ACL を持つバケットにオブジェクトをアップロードできます。

```
aws s3api put-object --bucket DOC-EXAMPLE-BUCKET --key key-name --body path-to-file --
acl bucket-owner-full-control
```

以下の PutObject オペレーションは ACL を指定しないため、無効な ACL を持つバケットでも成功します。

```
aws s3api put-object --bucket DOC-EXAMPLE-BUCKET --key key-name --body path-to-file
```

> **Note**
>
> アップロード後に他の AWS アカウント が、オブジェクトにアクセスする必要がある場合は、バケットポリシーを介してそれらのアカウントに追加のアクセス許可を付与する必要があります。詳細については、「チュートリアル: ポリシーを使用した Amazon S3 リソースへのアクセスの管理」を参照してください。

ACL の再有効化

[バケット所有者の強制] 設定から別のオブジェクト所有権設定にいつでも変更することで ACL を再度有効にできます。[バケット所有者の強制] 設定を適用する前に、許可管理のオブジェクト ACL を使用し、これらのオブジェクト ACL アクセス権限をバケットポリシーに移行しなかった場合は、ACL を再度有効にした後に、これらのアクセス許可が復元されます。さらに、[バケット所有者の強制] 設定が適用されている間にバケットに書き込まれたオブジェクトは、バケット所有者によって所有されます。

例えば、[バケット所有者の強制] から [オブジェクトライター] 設定に戻すと、バケット所有者として、他の AWS アカウント が以前に所有していたオブジェクトを所有できなくなり、完全に制御できなくなります。代わりに、アップロードするアカウントがこれらのオブジェクトを再び所有します。他のアカウントが所有するオブジェクトは、許可用の ACL を使用するため、これらのオブジェクトに許可を与えるためポリシーを使用できません。ただし、[バケット所有者の強制] 設定が適用さ

れている間、バケットに書き込まれたオブジェクトは、バケット所有者として引き続き所有されます。ACL を再度有効にしても、これらのオブジェクトはオブジェクトライターによって所有されません。

AWS Management Console、AWS Command Line Interface (CLI)、REST API、または AWS SDK を使用して ACL を有効化および管理する手順については、ACL の設定 を参照してください。

ACL を無効にする前提条件

既存のバケットの ACL を無効にする前に、以下の前提条件を満たしていることを確認してください。

バケットとオブジェクト ACL を確認し、ACL アクセス権限を移行します。

ACL を無効にすると、バケットおよびオブジェクト ACL によって付与される許可がアクセスに影響しなくなります。ACL を無効にする前に、バケットとオブジェクト ACL を確認します。

バケット ACL がアカウント外の他のユーザーに読み取りまたは書き込みの許可を付与する場合は、[バケット所有者の強制] 設定を適用する前に、これらの許可をバケットポリシーに移行する必要があります。アカウントの外部で読み取りまたは書き込みアクセスを許可するバケット ACL を移行しないと、[バケット所有者の強制] 設定を適用するリクエストが失敗し、InvalidBucketAclWithObjectOwnership エラーコードを返します。

例えば、サーバーアクセスログを受信するバケットの ACL を無効にするには、S3 ログ配信グループのバケット ACL アクセス権限をバケットポリシーのログサービスプリンシパルに移行する必要があります。詳細については、「サーバーアクセスのログ記録用の S3 ログ配信グループへのアクセスを付与する」を参照してください。

オブジェクトライターがアップロードするオブジェクトを完全に制御できるようにするには、オブジェクトライターはユースケースに最適なオブジェクト所有者設定です。個々のオブジェクトレベルでアクセスを制御する場合は、バケット所有者が優先するのが最適です。これらのユースケースはまれです。

ACL を確認して ACL の許可をバケットポリシーに移行するには、ACL を無効にする前提条件 を参照してください。

承認に ACL を必要としたリクエストを特定する

承認に ACL を必要とした Amazon S3 リクエストを特定するには、Amazon S3 サーバーアクセスログまたは AWS CloudTrail の `aclRequired` 値を使用します。リクエストが承認に ACL を必要としたか、ACL を指定する PUT リクエストがある場合、文字列は `Yes` です。ACL が不要であった

か、bucket-owner-full-control の既定 ACL を設定するか、リクエストがバケットポリシーで許可されている場合、aclRequired 値の文字列は Amazon S3 サーバーアクセスログでは「-」となり、CloudTrail では存在しません。期待される aclRequired 値の詳細については、「一般的な Amazon S3 リクエストの aclRequired 値」を参照してください。

PutBucketAcl または PutObjectAcl リクエストに ACL ベースのアクセス許可を付与するヘッダーが含まれている場合は、bucket-owner-full-control 既定 ACL を除き、ACL を無効にする前にこれらのヘッダーを削除する必要があります。待機しないと、リクエストは失敗します。

承認に ACL を必要としたその他すべてのリクエストについては、これらの ACL アクセス許可をバケットポリシーに移行します。次に、バケット所有者の強制設定を有効にする前に、バケットの ACL をすべて削除します。

> **Note**
>
> オブジェクトの ACL は削除しないでください。削除すると、アクセス許可をオブジェクトの ACL に依存しているアプリケーションはアクセスできなくなります。

承認に ACL を必要としたリクエストがない場合は、ACL の無効化に進むことができます。リクエストの識別方法の詳細については、「Amazon S3 アクセスログを使用したリクエストの識別」と「CloudTrail を使用した Amazon S3 リクエストの識別」を参照してください。

ACL 関連の条件キーを使用するバケットポリシーを確認および更新します。

[バケット所有者の強制] 設定を適用して ACL を無効にすると、リクエストでバケット所有者の完全制御 ACL が使用されているか、ACL が指定されていない場合にのみ、新しいオブジェクトをバケットにアップロードできます。ACL を無効にする前に、ACL 関連の条件キーのバケットポリシーを確認します。

バケットポリシーが ACL 関連の条件キーを使用して、bucket-owner-full-control 既定 ACL (例えば、s3:x-amz-acl) では、バケットポリシーを更新する必要はありません。次のバケットポリシーでは、s3:x-amz-acl S3 の規定 ACL bucket-owner-full-control リクエストを必要とする PutObject を使用します。このポリシーでは、引き続きオブジェクトライターが bucket-owner-full-control 既定 ACL を指定する必要があります。ただし、ACL が無効になっているバケットは引き続きこの ACL を受け入れるため、クライアント側の変更は不要で、リクエストは引き続き成功します。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Only allow writes to my bucket with bucket owner full control",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/ExampleUser"
                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

ただし、バケットポリシーで異なる ACL を必要とする ACL 関連の条件キーを使用する場合は、この条件キーを削除する必要があります。この例のバケットポリシーには、 S3 の public-read ACL を PutObject リクエストして、ACL が無効になる前に更新する必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Only allow writes to my bucket with public read access",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/ExampleUser"                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "StringEquals": {
```

```
                    "s3:x-amz-acl": "public-read"
                }
            }
        }
    ]
}
```

オブジェクトの所有者許可

バケットのオブジェクト所有権設定を適用、更新、または削除するには、`s3:PutBucketOwnershipControls` 許可が必要です。バケットのオブジェクトの所有権の設定を返すには、`s3:GetBucketOwnershipControls` 許可が必要です。詳細については、バケットの作成時のオブジェクトの所有権の設定。およびS3 バケットのオブジェクト所有権設定の表示を参照してください。

すべての新しいバケットの ACL を無効にします。

デフォルトでは、新しく作成されたバケットにはすべて[バケット所有者の強制] 設定が適用されており、ACL は無効になっています。ACL は無効にしておくことをお勧めします。原則として、アクセスコントロールには ACL ではなく S3 リソースベースのポリシー (バケットポリシーとアクセスポイントポリシー) または IAM ポリシーを使用することをお勧めします。ポリシーとは、よりシンプルで柔軟なアクセスコントロールのオプションです。バケットポリシーとアクセスポイントポリシーを使用すると、Amazon S3 リソースに対するすべてのリクエストに広く適用されるルールを定義できます。

レプリケーションとオブジェクトの所有権。

S3 レプリケーションを使用し、ソースバケットと宛先バケットが異なる AWS アカウント に所有されている場合、ACL を無効にして (バケット所有者にオブジェクト所有権の設定を適用して)、レプリカの所有権を宛先バケットを所有する AWS アカウント に変更できます。この設定は、`s3:ObjectOwnerOverrideToBucketOwner` 許可の必要なく、既存の所有者のオーバーライド動作を模倣します。[バケット所有者の強制] 設定で宛先バケットにレプリケーションされるすべてのオブジェクトは、宛先バケット所有者によって所有されます。レプリケーション設定の所有者上書きオプションの詳細については、レプリカ所有者の変更 を参照してください。

オブジェクトの所有権の設定

オブジェクト所有権の設定は、S3 コンソール、AWS CLI、AWS SDK、Amazon S3 REST API、または AWS CloudFormation を使用して適用できます。以下の REST API と AWS CLI コマンドは、オブジェクトの所有権をサポートしています。

REST API	AWS CLI	説明
PutBucketOwnershipControls	put-bucket-ownership-controls	既存の S3 バケットのオブジェクト所有権設定を作成または変更します。
CreateBucket	create-bucket	`x-amz-object-ownership` リクエストヘッダーを使用してバケットを作成し、オブジェクトの所有権設定を指定します。
GetBucketOwnershipControls	get-bucket-ownership-controls	Amazon S3 バケットのオブジェクト所有権設定を取り戻します。
DeleteBucketOwnershipControls	delete-bucket-ownership-controls	Amazon S3 バケットのオブジェクトの所有権の設定を削除します。

オブジェクト所有権設定の適用と操作の詳細については、以下のトピックを参照してください。

トピック

- ACL を無効にする前提条件
- バケットの作成時のオブジェクトの所有権の設定。
- 既存のバケットでのオブジェクトの所有権の設定
- S3 バケットのオブジェクト所有権設定の表示
- すべての新しいバケットの ACL を無効にし、オブジェクト所有権を執行します。
- トラブルシューティング

ACL を無効にする前提条件

バケット ACL が AWS アカウント の外部へのアクセスを許可する場合は、ACL を無効にする前にバケット ACL アクセス権限をバケットポリシーに移行し、バケット ACL をデフォルトのプライベート ACL にリセットする必要があります。これらのバケット ACL を移行

しない場合、ACL を無効にするために [バケット所有者の強制] 設定を適用する要求は失敗し、InvalidBucketAclWithObjectOwnership エラーコードが返されます。また、オブジェクト ACL アクセス権限を確認し、バケットポリシーに移行することをお勧めします。推奨されるその他の前提条件の詳細については、ACL を無効にする前提条件 を参照してください。

IAM ポリシーでは、既存のバケットとオブジェクト ACL にはそれぞれ同等のものが含まれます。次のバケットポリシー例は、バケットおよびオブジェクト ACL の READ と WRITE の許可は IAM 許可にマッピングされます。各 ACL が IAM アクセス許可に変換される方法の詳細については、ACL アクセス許可とアクセスポリシーのアクセス許可のマッピング を参照してください。

バケットポリシーへの ACL アクセス権限を確認し、移行するには、以下のトピックを参照してください。

トピック

- バケットポリシーの例
- S3 コンソールを使用した、ACL アクセス権限の確認と移行
- AWS CLI を使用して、ACL アクセス権限を確認して移行します。
- チュートリアル例

バケットポリシーの例

これらのバケットポリシーの例は、READ と WRITE バケット、サードパーティー AWS アカウントの ACL アクセス権限をバケットポリシーに移行およびバケット化およびオブジェクト化する方法を示しています。READ_ACP と WRITE_ACP ACL は、ACL 関連のアクセス許可 (s3:GetBucketAcl、s3:GetObjectAcl、s3:PutBucketAcl、s3:PutObjectAcl) を付与するため、ポリシーとの関連性は低くなります。

Example — バケットの **READ** ACL

バケットの内容の一覧表示を AWS アカウント *111122223333* に許可する READ ACL がバケットにある場合、バケットに対して s3:ListBucket、s3:ListBucketVersions、s3:ListBucketMultipartUploads の各アクセス許可を付与するバケットポリシーを記述できます。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "Permission to list the objects in a bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": [

                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:ListBucket",
                "s3:ListBucketVersions",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
        }
    ]
}
```

Example — バケット内の各オブジェクトの **READ** ACL

AWS アカウント *111122223333* にアクセスを許可する READ ACL がバケット内の各オブジェクトにある場合、バケット内の各オブジェクトに対する s3:GetObject アクセス許可と s3:GetObjectVersion アクセス許可をこのアカウントに付与するバケットポリシーを記述できます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Read permission for every object in a bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
```

```
    ]
}
```

この例のリソースエレメントは、特定のオブジェクトへのアクセスを許可します。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/OBJECT-KEY"
```

Example — バケットにオブジェクトを書き込むアクセス許可を付与する **WRITE** ACL

バケットへのオブジェクトの書き込み許可を AWS アカウント *111122223333* に付与する WRITE ACL がバケットにある場合は、バケットに s3:PutObject 許可を付与するバケットポリシーを記述できます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Permission to write objects to a bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

S3 コンソールを使用した、ACL アクセス権限の確認と移行

バケットの ACL アクセス権限を確認する方法

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、バケット名を選択します。
3. [アクセス許可] タブを選択します。

4. アクセスコントロールリスト (ACL) で、バケット ACL アクセス権限を確認します。

オブジェクトの ACL アクセス権限を確認する方法

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets] (バケット) リストで、オブジェクトが含まれるバケットの名前を選択します。
3. [オブジェクト] リストで、オブジェクト名を選択します。
4. [アクセス許可] タブを選択します。
5. アクセスコントロールリスト (ACL) で、オブジェクト ACL アクセス権限を確認します。

ACL アクセス権限を移行してバケット ACL を更新します

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、バケット名を選択します。
3. アクセス許可タブの バケットポリシーで 編集をクリックします。
4. ポリシーボックスで、バケットポリシーを追加または更新します。

   バケットポリシーの例ついては、バケットポリシーの例」 と「チュートリアル例 を参照してください。
5. [Save changes] (変更の保存) をクリックします。
6. バケット ACL を更新して、他のグループ、または AWS アカウント への ACL アクセス権限を削除します。
7. オブジェクト所有権にバケット所有者の強制設定を適用します。

AWS CLI を使用して、ACL アクセス権限を確認して移行します。

1. バケットのバケット ACL を返すには、get-bucket-acl AWS CLI コマンドを使用します。

   ```
   aws s3api get-bucket-acl --bucket DOC-EXAMPLE-BUCKET
   ```

   例えば、このバケット ACL は WRITE と READ にサードパーティーのアカウントへのアクセスを付与します。この ACL では、サードパーティーアカウントは正規ユーザー ID で識別しま

す。[バケット所有者の強制] 設定を適用して ACL を無効にするには、サードパーティアカウントのこれらの許可をバケットポリシーに移行する必要があります。

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "DisplayName": "THIRD-PARTY-EXAMPLE-ACCOUNT",
                "ID":
 "72806de9d1ae8b171cca9e2494a8d1335dfced4ThirdPartyAccountCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "READ"
        },
        {
            "Grantee": {
                "DisplayName": "THIRD-PARTY-EXAMPLE-ACCOUNT",
                "ID":
 "72806de9d1ae8b171cca9e2494a8d1335dfced4ThirdPartyAccountCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "WRITE"
        }
    ]
}
```

その他の ACL の例については、チュートリアル例 を参照してください。

2. バケット ACL アクセス権限をバケットポリシーに移行します。

この例のバケットポリシーは、サードパーティアカウントの s3:PutObject と s3:ListBucket アクセス許可を付与します。バケットポリシーでは、サードパーティーアカウントは AWS アカウント ID (*111122223333*)で識別されます。

```
aws s3api put-bucket-policy --bucket DOC-EXAMPLE-BUCKET --policy file://policy.json

policy.json:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PolicyForCrossAccountAllowUpload",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": [
             "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
             "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

バケットポリシーの例については、バケットポリシーの例 と チュートリアル例 を参照してください。

3. 特定のオブジェクトの ACL を返すには、get-object-acl AWS CLI コマンドを使用します。

```
aws s3api get-object-acl --bucket DOC-EXAMPLE-BUCKET --key EXAMPLE-OBJECT-KEY
```

4. 必要に応じて、オブジェクト ACL アクセス権限をバケットポリシーに移行します。

   このリソースエレメントの例では、バケットポリシーの特定のオブジェクトへのアクセスを付与します。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/EXAMPLE-OBJECT-KEY"
```

5. バケットの ACL をデフォルトの ACL にリセットします。

```
aws s3api put-bucket-acl --bucket DOC-EXAMPLE-BUCKET --acl private
```

6. オブジェクト所有者に[バケット所有者の強制] 設定を適用します。

チュートリアル例

以下の例では、特定のユースケースでの ACL アクセス権限をバケットポリシーに移行する方法を示します。

トピック

- サーバーアクセスのログ記録用の S3 ログ配信グループへのアクセスを付与する
- バケット内のオブジェクトへのパブリック読み取りアクセスを付与します
- S3 バケットに Amazon ElastiCache for Redis アクセスを付与する

サーバーアクセスのログ記録用の S3 ログ配信グループへのアクセスを付与する

バケット所有者の強制設定を適用してサーバーアクセスログ送信先バケット (別名 ターゲットバケット) の ACL を無効にするには、バケットの S3 ログ配信グループへの ACL アクセス許可をバケットポリシーのログ記録サービスプリンシパル (logging.s3.amazonaws.com) に移行する必要があります。ログ配信許可の詳細については、ログ配信許可 を参照してください。

このバケット ACL は、S3 ログ配信グループへの WRITE と READ_ACP アクセスを許可します。

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "Type": "CanonicalUser",
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
```

```
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery"
            },
            "Permission": "WRITE"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery"
            },
            "Permission": "READ_ACP"
        }
    ]
}
```

S3 ログ配信グループのバケット ACL アクセス権限を、バケットポリシーのロギングサービスプリンシパルに移行します。

1. 次のバケットポリシーをバケットに追加して、例の値を置き換えます。

```
aws s3api put-bucket-policy --bucket DOC-EXAMPLE-BUCKET --policy file://policy.json

policy.json:      {
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/EXAMPLE-LOGGING-PREFIX*",
            "Condition": {
                "ArnLike": {
```

```
                    "aws:SourceArn": "arn:aws:s3:::SOURCE-BUCKET-NAME"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

2. バケットの ACL をデフォルトの ACL にリセットします。

```
aws s3api put-bucket-acl --bucket DOC-EXAMPLE-BUCKET --acl private
```

3. オブジェクト所有権のバケット所有者の強制 を送信先バケットに適用します。

バケット内のオブジェクトへのパブリック読み取りアクセスを付与します

オブジェクト ACL がバケット内のすべてのオブジェクトへのパブリック読み取りアクセスを許可する場合、これらの ACL アクセス権限をバケットポリシーに移行できます。

このオブジェクト ACL は、バケット内のオブジェクトへのパブリック読み取りアクセスを許可します。

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
```

```
            },
            "Permission": "READ"
        }
    ]
}
```

パブリック読み取り ACL アクセス権限をバケットポリシーに移行します。

1. バケット内のすべてのオブジェクトへのパブリック読み取りアクセスを許可するには、例の値を置き換えて、次のバケットポリシーを追加します。

```
aws s3api put-bucket-policy --bucket DOC-EXAMPLE-BUCKET --policy file://policy.json

policy.json:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        }
    ]
}
```

バケットポリシー内の特定のオブジェクトへのパブリックアクセスを許可するには、Resource エレメントに次の形式を使用します。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/OBJECT-KEY"
```

特定のプレフィックスを持つすべてのオブジェクトへのパブリックアクセスを許可するには、Resource エレメントに次の形式を使用します。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/PREFIX/*"
```

2. オブジェクト所有者に[バケット所有者の強制] 設定を適用します。

S3 バケットに Amazon ElastiCache for Redis アクセスを付与する

ElastiCache for Redis バックアップを S3 バケットにエクスポートできます。これにより、ElastiCache の外部からバックアップにアクセスできます。バックアップを S3 バケットにエクスポートするには、スナップショットをバケットにコピーするための ElastiCache 許可を付与する必要があります。バケット ACL で ElastiCache にアクセス許可を付与した場合は、[バケット所有者の強制] 設定を適用して ACL を無効にする前に、これらの許可をバケットポリシーに移行する必要があります。詳細については、Amazon ElastiCache ユーザーガイド の Amazon S3 バケットへの ElastiCache アクセスの許可 を参照してください。

次の例は、ElastiCache に許可を付与するバケット ACL アクセス許可を示しています。

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "READ"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
```

```
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "WRITE"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "READ_ACP"
        }
    ]
}
```

ElastiCache forRedis のバケット ACL アクセス権限をバケットポリシーに移行します。

1. 次のバケットポリシーをバケットに追加し、例の値を置き換えます。

```
aws s3api put-bucket-policy --bucket DOC-EXAMPLE-BUCKET --policy file://policy.json

policy.json:
"Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "Region.elasticache-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
```

```
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        }
    ]
}
```

2. バケットの ACL をデフォルトの ACL にリセットします。

```
aws s3api put-bucket-acl --bucket DOC-EXAMPLE-BUCKET --acl private
```

3. オブジェクト所有者に[バケット所有者の強制] 設定を適用します。

バケットの作成時のオブジェクトの所有権の設定。

バケットを作成するときに、S3 オブジェクトの所有権を設定できます。既存のバケットのオブジェクトの所有権を設定するには、既存のバケットでのオブジェクトの所有権の設定 を参照してください。

S3 オブジェクトの所有権は、アクセスコントロールリスト (ACL) を無効化するために使用できる Amazon S3 バケットレベルの設定で、バケット内のすべてのオブジェクトの所有権を取得し、Amazon S3 に保存されているデータのアクセス管理を簡素化します。デフォルトでは、オブジェクト所有者はバケット所有者の強制設定に設定され、新しいバケットの ACL が無効になります。ACL を無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。

オブジェクト所有権には、バケットにアップロードされるオブジェクトの所有権を制御し、ACL を無効または有効化するために使用できる 3 つの設定があります。

ACL は無効です

- バケット所有者強制 (デフォルト) – ACL は無効になり、バケット所有者はバケット内のすべてのオブジェクトを自動的に所有し、完全に制御します。ACL は、S3 バケット内のデータに対するアクセス許可に対して影響を与えません。バケットは、ポリシーを使用してアクセスコントロールを定義します。

ACL の有効化

- 希望するバケット所有者 — バケット所有者は、他のアカウントが `bucket-owner-full-control` 既定 ACL でバケットに書き込む新しいオブジェクトを所有し、完全にコントロールできます。
- オブジェクトライター — オブジェクトをアップロードする AWS アカウント は、そのオブジェクトを所有し、そのオブジェクトを完全にコントロールし、ACL を介して他のユーザーにそのオブジェクトへのアクセス権を付与できます。

アクセス許可: [Bucket owner enforced] (バケット所有者の強制) 設定、または [Bucket owner preferred] (バケット所有者の優先) 設定を適用するには、`s3:CreateBucket` および `s3:PutBucketOwnershipControls` のアクセス許可が必要です。オブジェクトライター設定が適用されたバケットを作成する場合、追加の権限は必要ありません。詳細については、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

> **Important**
>
> Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなり、オブジェクトごとに個別にアクセスを制御する必要がある異常な状況を除き、ACL を無効にすることをお勧めします。オブジェクトの所有権を使用すると ACL を無効にして、アクセスコントロールに関するポリシーに依存できます。ACL を無効にすると、別の AWS アカウントによってアップロードされたオブジェクトを含むバケットを簡単に維持できます。バケット所有者は、バケット内のすべてのオブジェクトを所有し、ポリシーを使用してオブジェクトへのアクセスを管理できます。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ページ上部にあるナビゲーションバーで、現在表示されている AWS リージョン の名前をクリックします。次に、バケットを作成するリージョンを選択します。

   > **Note**
   >
   > レイテンシーとコストを最小化するため、さらに規制条件に対応するために、最寄りのリージョンを選択します。明示的に別のリージョンに移動する場合を除き、特定のリー

ジョンに保管されたオブジェクトは、そのリージョンから移動されることはありません。Amazon S3 AWS リージョン のリストについては、「Amazon Web Services 全般のリファレンス」の「AWS のサービス エンドポイント」を参照してください。

3. 左側のナビゲーションペインで、[バケット] を選択します。
4. [Create bucket (バケットの作成)] を選択します。

   [バケットの作成] ページが開きます。
5. [全般設定] で、バケットが作成される AWS リージョン を確認します。
6. [バケットタイプ] で、[汎用] を選択します。
7. [バケット名] にバケットの名前を入力します。

   バケット名には次の条件があります。

   - パーティション内で一意にする必要があります。パーティションは、リージョンのグループです。AWS には、現在、aws (標準リージョン)、aws-cn (中国リージョン)、および aws-us-gov (AWS GovCloud (US) Regions) の 3 つのパーティションがあります。
   - 3 ~ 63 文字で指定する。
   - 小文字、数字、ドット (.)、およびハイフン (-) のみで構成できます。互換性を最も高くするには、静的ウェブサイトホスティング専用のバケットを除き、バケット名にドット (.) を使用しないことをお勧めします。
   - 文字や数字で始まり、文字や数字で終わります。

   バケットを作成したら、その名前を変更することはできません。バケットの命名の詳細については、「バケットの名前付け」を参照してください。

   > **Important**
   >
   > バケット名にアカウント番号などの機密情報を含めないでください。バケット名は、バケット内のオブジェクトを参照する URL に表示されます。

8. AWS Management Console では、既存のバケットの設定を新しいバケットにコピーできます。既存のバケットの設定をコピーしない場合は、次のステップにスキップします。

   > **Note**
   >
   > このオプションの特徴:

- AWS CLI では使用できません。コンソールでのみ利用できます。
- ディレクトリバケットには利用できません。
- バケットポリシーは既存のバケットから新しいバケットにコピーしません。

既存のバケットの設定をコピーするには、[既存のバケットから設定をコピー] で [バケットを選択] をクリックします。[バケットを選択] ウィンドウが開きます。コピーする設定を持つバケットを検索して、[バケットを選択] をクリックします。[バケットを選択] ウィンドウが閉じて、[バケットを作成] ウィンドウが再び開きます。

[既存のバケットから設定をコピー] に、選択したバケットの名前が表示されるようになります。コピーしたバケット設定を削除するための [デフォルトを復元] オプションも表示されます。[バケットを作成] ページで、バケットの残りの設定を確認します。ここで、選択したバケットの設定と一致していることを確認できます。最後のステップにスキップできます。

9. [オブジェクト所有者] で、ACL を無効または有効にし、バケットにアップロードされたオブジェクトの所有権を制御するには、次のいずれかの設定を選択します。

   ACL は無効です

   - バケット所有者強制 (デフォルト) – ACL は無効になり、バケット所有者はバケット内のすべてのオブジェクトを自動的に所有し、完全に制御します。ACL は、S3 バケット内のデータに対するアクセス許可に対して影響を与えません。このバケットはアクセスコントロールを定義するためだけにポリシーを使用します。

     デフォルトでは、ACL は無効になっています。Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

   ACL の有効化

   - 希望するバケット所有者 — バケット所有者は、他のアカウントが `bucket-owner-full-control` 既定 ACL でバケットに書き込む新しいオブジェクトを所有し、完全にコントロールできます。

[バケット所有者を推奨] 設定を適用して、すべての Amazon S3 アップロードに `bucket-owner-full-control` 既定 ACL を含めることを要求する場合は、この ACL を使用するオブジェクトアップロードのみを許可するバケットポリシーを追加できます。

- オブジェクトライター — オブジェクトをアップロードする AWS アカウント は、そのオブジェクトを所有し、そのオブジェクトを完全にコントロールし、ACL を介して他のユーザーにそのオブジェクトへのアクセス権を付与できます。

> **Note**
>
> デフォルト設定は [バケット所有者の強制] です。デフォルト設定を適用して ACL を無効のままにするのに必要なのは、`s3:CreateBucket` アクセス許可のみです。ACL を有効にするためには、`s3:PutBucketOwnershipControls` アクセス許可が必要です。

10. [このバケットのパブリックアクセスブロック設定] で、バケットに適用するブロックパブリックアクセス設定を選択します。

    デフォルトでは、4 つすべての [パブリックアクセスをブロック] 設定が有効になっています。特定のユースケースでオフにする必要のある設定が 1 つ以上あることがわかっている場合を除き、すべての設定を有効にしておくことをお勧めします。ブロックパブリックアクセスの詳細については、Amazon S3 ストレージへのパブリックアクセスのブロック を参照してください。

    > **Note**
    >
    > すべての [パブリックアクセスをブロック] 設定を有効にするのに必要なのは、`s3:CreateBucket` アクセス許可のみです。[パブリックアクセスをブロック] 設定をオフにするには、`s3:PutBucketPublicAccessBlock` アクセス許可が必要です。

11. (オプション) [バケットのバージョニング] では、オブジェクトのバリアントをバケットに保持するかどうかを選択できます。バージョニングの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

    バケットのバージョニングを無効または有効にするには、[Disable] (無効化) または [Enable] (有効化) を選択します。

12. (オプション) [Tags] (タグ) では、バケットにタグを追加することを選択できます。タグは、ストレージの分類に使用されるキーと値のペアです。

(オプション) タグを追加するには、[キー] を入力してから、オプションの [値] を入力し、[タグの追加] を選択します。

13. [デフォルトの暗号化] で、[編集] を選択します。
14. デフォルトの暗号化を設定するには、[暗号化タイプ] で次のいずれかを選択します。

    - Amazon S3 マネージドキー (SSE-S3)
    - AWS Key Management Service キー (SSE-KMS)

    > **Important**
    >
    > デフォルト暗号化設定に SSE-KMS オプションを使用する場合、AWS KMS の 1 秒あたりのリクエスト (RPS) 制限が適用されます。AWS KMS クォータの詳細およびクォータの引き上げをリクエストする方法については、「AWS Key Management Service デベロッパーガイド」の「クォータ」を参照してください。

    バケットと新しいオブジェクトは、暗号化設定の基本レベルとして Amazon S3 マネージドキーを使用したサーバー側の暗号化で暗号化されます。デフォルトの暗号化の詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。

    Amazon S3 のサーバー側の暗号化を使用してデータを暗号化する方法の詳細については、「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)」を参照してください。
15. [AWS Key Management Service キー (SSE-KMS)] を選択した場合は、以下の操作を実行します。

    a. [AWS KMS キー] で、次のいずれかの方法で KMS キーを指定します。

        - 使用可能な KMS キーのリストから選択するには、[AWS KMS keys から選択する] を選択し、使用可能なキーのリストから自分の KMS キーを選択します。

          AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がこのリストに表示されます。カスタマーマネージドキーの詳細については、AWS Key Management Service デベロッパーガイドの「カスタマーキーと AWS キー」を参照してください。
        - KMS キー ARN を入力するには、[AWS KMS key ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。

- AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

  AWS KMS key の作成の詳細については、AWS Key Management Service デベロッパーガイドのキーの作成を参照してください。

> **Important**
>
> バケットと同じ AWS リージョン で使用可能な KMS キーのみを使用できます。Amazon S3 コンソールには、バケットと同じリージョンで最初の 100 個の KMS キーしか表示されません。リストに存在しない KMS キーを使用するには、KMS キー ARN を入力する必要があります。別のアカウントが所有している KMS キーを使用する場合は、まずそのキーを使用するアクセス許可が必要であり、次に KMS キー ARN を入力する必要があります。KMS キーのクロスアカウント権限の詳細については、AWS Key Management Service デベロッパーガイドの「他のアカウントで使用できる KMS キーを作成する」を参照してください。SSE-KMS に関する詳細は、「AWS KMS (SSE-KMS) によるサーバー側の暗号化の指定」を参照してください。
> Amazon S3 でサーバー側の暗号化に AWS KMS key を使用する場合は、対称暗号化 KMS キーを選択する必要があります。Amazon S3 では、対称暗号化 KMS キーのみがサポートされ、非対称暗号化 KMS キーはサポートされません。詳細については、AWS Key Management Service デベロッパーガイドの「Identifying symmetric and asymmetric KMS keys」(対称および非対称 KMS キーの識別) を参照してください。

AWS KMS key の作成の詳細については、AWS Key Management Service デベロッパーガイドのキーの作成を参照してください。Amazon S3 での AWS KMS の使用に関する詳細は、「AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用」を参照してください。

b. SSE-KMS でデフォルトの暗号化を使用するようにバケットを設定する場合は、S3 バケットキーを有効にすることもできます。S3 バケットキーは、Amazon S3 から AWS KMS へのリクエストトラフィックを減らし、暗号化のコストを削減します。詳細については、「Amazon S3 バケットキーを使用した SSE-KMS のコストの削減」を参照してください。

   S3 バケットキーを使用するには、[バケットキー] で [有効化] を選択します。

16. (オプション) S3 オブジェクトロックを有効にする場合は、次の手順に従います。

    a. [詳細設定] を選択します。

    > **Important**
    >
    > バケットに対してオブジェクトロックを有効にすると、バージョニングも有効になります。有効にした後、オブジェクトロックのデフォルト保持設定およびリーガルホールド設定を指定し、新しいオブジェクトを削除または上書きしないようにする必要があります。

    b. オブジェクトロックを有効にする場合は、[Enable] (有効化) を選択し、表示される警告を読んだうえで承認します。

    詳細については、「S3 オブジェクトロックの使用」を参照してください。

    > **Note**
    >
    > オブジェクトロックが有効なバケットを作成するには、`s3:CreateBucket`、`s3:PutBucketVersioning` および `s3:PutBucketObjectLockConfiguration` の許可が必要です。

17. [Create bucket (バケットの作成)] を選択します。

AWS CLI の使用

新しいバケットを作成するときにオブジェクトの所有権を設定するには、`create-bucket` パラメータを指定して AWS CLI と `--object-ownership` コマンドを使用します。

この例では、AWS CLI を使用して新しいバケットに [バケット所有者の強制] 設定を適用します。

```
aws s3api create-bucket --bucket  DOC-EXAMPLE-BUCKET --region us-east-1 --object-
ownership BucketOwnerEnforced
```

> **Important**
>
> AWS CLI を使用してバケットを作成するときにオブジェクト所有権を設定しない場合、デフォルト設定は `ObjectWriter` (ACL 有効) になります。

AWS SDK for Java の使用

この例では、AWS SDK for Java を使用して新しいバケットの [バケット所有者の強制] 設定を設定します。

```
    // Build the ObjectOwnership for CreateBucket
    CreateBucketRequest createBucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectOwnership(ObjectOwnership.BucketOwnerEnforced)
            .build()

     // Send the request to Amazon S3
     s3client.createBucket(createBucketRequest);
```

AWS CloudFormation の使用

新しいバケットを作成するときに `AWS::S3::Bucket` と AWS CloudFormation リソースを使用してオブジェクトの所有権を設定するには、「AWS CloudFormation ユーザーガイド」の「AWS::S3::Bucket の OwnershipControls」を参照してください。

REST API の使用

S3 オブジェクト所有権に [バケット所有者の強制] 設定を適用するには、`x-amz-object-ownership` リクエストヘッダーを `BucketOwnerEnforced` に設定して `CreateBucket` API オペレーションを使用します。詳細と例については、「Amazon Simple Storage Service API リファレンス」の「CreateBucket」を参照してください。

次のステップ: オブジェクトの所有権にバケット所有者の強制設定またはバケット所有者の優先設定を適用した後、さらに次の手順を実行できます。

- バケット所有者の強制 – IAM または組織ポリシーを使用して、ACL を無効にしてすべての新しいバケットを作成する必要があります。
- バケット所有者推奨 – S3 バケットポリシーを追加して、バケットへのすべてのオブジェクトのアップロードに `bucket-owner-full-control` 規定 ACL を要求します

既存のバケットでのオブジェクトの所有権の設定

既存の S3 バケットで S3 オブジェクトの所有権を設定できます。バケットを作成するときにオブジェクトの所有権を適用するには、バケットの作成時のオブジェクトの所有権の設定。 を参照してください。

S3 オブジェクトの所有権は、アクセスコントロールリスト (ACL) を無効化するために使用できる Amazon S3 バケットレベルの設定で、バケット内のすべてのオブジェクトの所有権を取得し、Amazon S3 に保存されているデータのアクセス管理を簡素化します。デフォルトでは、オブジェクト所有者はバケット所有者の強制設定に設定され、新しいバケットの ACL が無効になります。ACL を無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。

オブジェクト所有権には、バケットにアップロードされるオブジェクトの所有権を制御し、ACL を無効または有効化するために使用できる 3 つの設定があります。

ACL は無効です

- バケット所有者強制 (デフォルト) – ACL は無効になり、バケット所有者はバケット内のすべてのオブジェクトを自動的に所有し、完全に制御します。ACL は、S3 バケット内のデータに対するアクセス許可に対して影響を与えません。バケットは、ポリシーを使用してアクセスコントロールを定義します。

ACL の有効化

- 希望するバケット所有者 — バケット所有者は、他のアカウントが `bucket-owner-full-control` 既定 ACL でバケットに書き込む新しいオブジェクトを所有し、完全にコントロールできます。
- オブジェクトライター — オブジェクトをアップロードする AWS アカウント は、そのオブジェクトを所有し、そのオブジェクトを完全にコントロールし、ACL を介して他のユーザーにそのオブジェクトへのアクセス権を付与できます。

前提条件: [バケット所有者の強制] 設定を適用して ACL を無効にする前に、バケット ACL アクセス権限をバケットポリシーに移行し、バケット ACL をデフォルトのプライベート ACL にリセットする必要があります。また、オブジェクト ACL アクセス権限をバケットポリシーに移行し、バケット所有者の完全制御 ACL 以外の ACL を必要とするバケットポリシーを編集することをお勧めします。詳細については、「ACL を無効にする前提条件」を参照してください。

許可: このオペレーションを使用するには、`s3:PutBucketOwnershipControls` 許可が必要です。詳細については、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. バケット リストで、S3 オブジェクトの所有者の設定を適用するバケットの名前を選択します。
3. [アクセス許可] タブを選択します。
4. オブジェクトの所有者で [編集] を選択します。
5. オブジェクト所有権は、ACL を無効または有効にして、バケットにアップロードされたオブジェクトの所有権を制御するために、次の設定のいずれか一つを選択します。

   ACL は無効です

   - バケット所有者の強制 - ACL は無効になり、バケット所有者はバケット内のすべてのオブジェクトを自動的に所有し、完全にコントロールできます。ACL は、S3 バケット内のデータに対するアクセス許可に対して影響を与えません。このバケットはアクセスコントロールを定義するためにポリシーを使用します。

     IAM または AWS Organizations ポリシーを使用し、ACL を無効にしてすべての新しいバケットを作成することを要求するには、すべての新しいバケットの ACL を無効にします (バケット所有者の強制) を参照してください。

   ACL の有効化

   - 希望するバケット所有者 — バケット所有者は、他のアカウントが `bucket-owner-full-control` 既定 ACL でバケットに書き込む新しいオブジェクトを所有し、完全にコントロールできます。

     バケット所有者の優先設定を適用して、すべての Amazon S3 アップロードに `bucket-owner-full-control` 既定 ACL を含めることを要求する場合は、この ACL を使用するオブジェクトアップロードのみを許可するバケットポリシーを追加できます。
   - オブジェクトライター — オブジェクトをアップロードする AWS アカウント は、そのオブジェクトを所有し、そのオブジェクトを完全にコントロールし、ACL を介して他のユーザーにそのオブジェクトへのアクセス権を付与できます。

6. [Save] を選択します。

AWS CLI の使用

既存のバケットにオブジェクトの所有権設定を適用するには、put-bucket-ownership-controls パラメータのある --ownership-controls コマンドを使用します。所有権の有効な値は BucketOwnerEnforced、BucketOwnerPreferred、または ObjectWriter です。

この例では、AWS CLI を使用して既存のバケットにバケット所有者の強制設定を適用します。

```
aws s3api put-bucket-ownership-controls --bucket DOC-EXAMPLE-BUCKET --ownership-
controls="Rules=[{ObjectOwnership=BucketOwnerEnforced}]"
```

put-bucket-ownership-controls の詳細については、「AWS Command Line Interface ユーザーガイド」の「put-bucket-ownership-controls」を参照してください。

AWS SDK for Java の使用

この例では、BucketOwnerEnforced を使用して既存のバケットに対してオブジェクト所有権の AWS SDK for Java 設定を適用します。

```
        // Build the ObjectOwnership for BucketOwnerEnforced
        OwnershipControlsRule rule = OwnershipControlsRule.builder()
               .objectOwnership(ObjectOwnership.BucketOwnerEnforced)
               .build();

        OwnershipControls ownershipControls = OwnershipControls.builder()
                  .rules(rule)
                  .build()

         // Build the PutBucketOwnershipControlsRequest
         PutBucketOwnershipControlsRequest putBucketOwnershipControlsRequest =
               PutBucketOwnershipControlsRequest.builder()
                       .bucket(BUCKET_NAME)
                       .ownershipControls(ownershipControls)
                       .build();

         // Send the request to Amazon S3
         s3client.putBucketOwnershipControls(putBucketOwnershipControlsRequest);
```

AWS CloudFormation の使用

既存のバケットにオブジェクト所有者の設定を適用するために AWS CloudFormation を使用するには、「AWS CloudFormation ユーザーガイド」の「AWS::S3::Bucket OwnershipControls」を参照してください。

REST API の使用

REST API を使用して既存の S3 バケットにオブジェクトの所有権設定を適用するには、`PutBucketOwnershipControls` を使用します。詳細については、「Amazon Simple Storage Service API リファレンス」の「PutBucketOwnershipControls」を参照してください。

次のステップ: オブジェクトの所有権にバケット所有者の強制設定またはバケット所有者の優先設定を適用した後、さらに次の手順を実行できます。

- バケット所有者の強制 – IAM または組織ポリシーを使用して、ACL を無効にしてすべての新しいバケットを作成する必要があります。
- バケット所有者推奨 – S3 バケットポリシーを追加して、バケットへのすべてのオブジェクトのアップロードに `bucket-owner-full-control` 規定 ACL を要求します

S3 バケットのオブジェクト所有権設定の表示

S3 オブジェクトの所有権は、アクセスコントロールリスト (ACL) を無効化するために使用できる Amazon S3 バケットレベルの設定で、バケット内のすべてのオブジェクトの所有権を取得し、Amazon S3 に保存されているデータのアクセス管理を簡素化します。デフォルトでは、オブジェクト所有者はバケット所有者の強制設定に設定され、新しいバケットの ACL が無効になります。ACL を無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。

オブジェクト所有権には、バケットにアップロードされるオブジェクトの所有権を制御し、ACL を無効または有効化するために使用できる 3 つの設定があります。

ACL は無効です

- バケット所有者強制 (デフォルト) – ACL は無効になり、バケット所有者はバケット内のすべてのオブジェクトを自動的に所有し、完全に制御します。ACL は、S3 バケット内のデータに対するアクセス許可に対して影響を与えません。バケットは、ポリシーを使用してアクセスコントロールを定義します。

ACL の有効化

- 希望するバケット所有者 — バケット所有者は、他のアカウントが `bucket-owner-full-control` 既定 ACL でバケットに書き込む新しいオブジェクトを所有し、完全にコントロールできます。
- オブジェクトライター — オブジェクトをアップロードする AWS アカウント は、そのオブジェクトを所有し、そのオブジェクトを完全にコントロールし、ACL を介して他のユーザーにそのオブジェクトへのアクセス権を付与できます。

Amazon S3 バケットの S3 オブジェクト所有権設定を表示できます。新しいバケットのオブジェクトの所有権を設定するには、バケットの作成時のオブジェクトの所有権の設定。 を参照してください。既存のバケットのオブジェクトの所有権を設定するには、既存のバケットでのオブジェクトの所有権の設定 を参照してください。

許可: このオペレーションを使用するには、`s3:GetBucketOwnershipControls` 許可が必要です。詳細については、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. バケットリストで、オブジェクト所有権設定を適用するバケットの名前を選択します。
3. [アクセス許可] タブを選択します。
4. オブジェクトの所有権 では、バケットのオブジェクト所有権設定を表示できます。

AWS CLI の使用

S3 バケットの S3 オブジェクト所有権設定を取得するには、get-bucket-ownership-controls AWS CLI コマンドを使用します。

```
aws s3api get-bucket-ownership-controls --bucket DOC-EXAMPLE-BUCKET
```

REST API の使用

S3 バケットのオブジェクトの所有権設定を検索するには、`GetBucketOwnershipControls` API オペレーションを使用します。詳細については、「GetBucketOwnershipControls」を参照してください。

すべての新しいバケットの ACL を無効にし、オブジェクト所有権を執行します。

Amazon S3 バケットで ACL を無効にすることをお勧めします。これを行うには、S3 オブジェクトの所有権に [バケット所有者の強制] 設定を適用します。この設定を適用すると、ACL が無効になり、バケット内のすべてのオブジェクトを自動的に所有し、完全に制御できるようになります。ACL を無効にしてすべての新しいバケットを作成するように要求するには、次のセクションで説明するように、AWS Identity and Access Management (IAM) ポリシーまたは AWS Organizations サービスコントロールポリシー (SCP) を使用します。

ACL を無効にせずに新しいオブジェクトのオブジェクト所有権を強制するには、バケット所有者の優先設定を適用できます。この設定を適用する場合は、バケットポリシーを更新して、バケットへのすべての PUT リクエストに `bucket-owner-full-control` 既定 ACL を要求することを強くお勧めします。`bucket-owner-full-control` 既定 ACL を他のアカウントのバケットに送信するように、クライアントも更新してください。

トピック

- すべての新しいバケットの ACL を無効にします (バケット所有者の強制)
- Amazon S3 PUT オペレーションで bucket-owner-full-control 既定 ACL を要求する (バケット所有者の優先)

すべての新しいバケットの ACL を無効にします (バケット所有者の強制)

次の例の IAM ポリシーは、[バケット所有者の強制] 設定がオブジェクト所有権に適用されていない限り、特定の IAM ユーザーまたはロールの `s3:CreateBucket` アクセス許可を拒否します。`Condition` ブロック内のキーバリューのペアは、キーには `s3:x-amz-object-ownership` を、値には `BucketOwnerEnforced` 設定を指定します。つまり、IAM ユーザーは、オブジェクト所有権に[バケット所有者の強制] 設定を設定し、ACL を無効にした場合にのみ、バケットを作成できます。このポリシーは、AWS 組織の境界 SCP として使用することもできます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireBucketOwnerFullControl",
            "Action": "s3:CreateBucket",
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
```

```
                    "s3:x-amz-object-ownership": "BucketOwnerEnforced"
                }
            }
        }
    ]
}
```

Amazon S3 **PUT** オペレーションで bucket-owner-full-control 既定 ACL を要求する (バケット所有者の優先)

オブジェクト所有権に対するバケット所有者の優先設定を使用すると、バケット所有者は、他のアカウントが bucket-owner-full-control 既定 ACL を使用してバケットに書き込む新しいオブジェクトを所有し、完全に制御できます。ただし、他のアカウントが bucket-owner-full-control 規定 ACL なしでバケットにオブジェクトを書き込む場合、オブジェクトライターは完全制御アクセスを維持します。バケット所有者は、bucket-owner-full-control 既定 ACL を指定した場合にのみ書き込みを許可するバケットポリシーを実装できます。

> **Note**
>
> [バケット所有者の強制] 設定で ACL を無効にすると、バケット所有者として、バケット内のすべてのオブジェクトを自動的に所有し、完全に制御できます。このセクションを使用してバケットポリシーを更新して、バケット所有者にオブジェクトの所有権を適用する必要はありません。

次のバケットポリシーでは、オブジェクトの ACL が bucket-owner-full-control に設定されている場合にのみ、アカウント *111122223333* が *DOC-EXAMPLE-BUCKET* にオブジェクトをアップロードできるように指定しています。お客様のアカウントで、*111122223333* を、お客様のバケット名で*DOC-EXAMPLE-BUCKET* を置き換えてください。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Only allow writes to my bucket with bucket owner full control",
         "Effect": "Allow",
         "Principal": {
            "AWS": [
               "arn:aws:iam::111122223333:user/ExampleUser"
            ]
```

```
         },
         "Action": [
            "s3:PutObject"
         ],
         "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
         "Condition": {
            "StringEquals": {
               "s3:x-amz-acl": "bucket-owner-full-control"
            }
         }
      }
   ]
}
```

次に、AWS Command Line Interface (AWS CLI) を使用した bucket-owner-full-control 既定 ACL を含むコピーオペレーションの例を示します。

```
aws s3 cp file.txt s3://DOC-EXAMPLE-BUCKET --acl bucket-owner-full-control
```

バケットポリシーが有効になった後、クライアントに bucket-owner-full-control 既定 ACL が含まれていない場合、オペレーションは失敗し、アップローダーは次のエラーを受け取ります。

PutObject オペレーションを呼び出すときにエラー (AccessDenied) が発生しました: アクセスが拒否されました。

> **Note**
>
> アップロード後にクライアントがオブジェクトにアクセスする必要がある場合は、アップロードアカウントに追加のアクセス権限を付与する必要があります。アカウントにリソースへのアクセス許可を付与する方法については、チュートリアル: ポリシーを使用した Amazon S3 リソースへのアクセスの管理 を参照してください。

トラブルシューティング

S3 オブジェクト所有権のバケット所有者強制設定を適用すると、アクセスコントロールリスト (ACL) が無効になり、バケット所有者として、バケット内のすべてのオブジェクトを自動的に所有することになります。ACL は、バケット内のオブジェクトの許可に影響を与えなくなりました。ポリシーを使用してアクセス許可を付与できます。すべての S3 PUT リクエストは、bucket-owner-full-control 既定 ACL を指定するか、ACL を指定しない必要があります。そうしないと失敗し

ます。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

無効な ACL が指定されているか、バケット ACL 許可が AWS アカウント の外部へのアクセスを許可している場合、次のエラー応答が表示されることがあります。

AccessControlListNotSupported

オブジェクト所有権に [バケット所有者の強制] 設定を適用すると、ACL は無効になります。ACL の設定または ACL の更新の要求は 400 エラーで失敗し、AccessControlListNotSupported エラーコードを返します。ACL の読み取り要求は引き続きサポートされています。ACL の読み取りリクエストは、バケット所有者の完全制御を示すレスポンスを常に返します。PUT オペレーションでは、バケット所有者の完全制御 ACL を指定するか、ACL を指定しないことが必要です。それ以外の場合、PUT オペレーションは失敗します。

次の put-object AWS CLI コマンドの例には、public-read 既定 ACL が含まれています。

```
aws s3api put-object --bucket DOC-EXAMPLE-BUCKET --key object-key-name --body doc-example-body --acl public-read
```

バケットが [バケット所有者の強制] 設定を使用して ACL を無効にする場合、このオペレーションは失敗し、アップローダーは次のエラーメッセージを受け取ります。

PutObject オペレーションの呼び出し時にエラー (AccessControlListNotSupported) が発生しました。バケットは ACL を許可していません

InvalidBucketAclWithObjectOwnership

[バケット所有者の強制] 設定を適用して ACL を無効にする場合、バケット ACL はバケット所有者にのみ完全制御を与える必要があります。バケット ACL は、外部 AWS アカウント または他のグループへのアクセスを許可できません。例えば、CreateBucket リクエストでバケット所有者の強制を設定し、外部 AWS アカウント へのアクセスを提供するバケット ACL を指定した場合、リクエストは 400 エラーで失敗し、InvalidBucketAclWithObjectOwnership エラーコードを返します。同様に、PutBucketOwnershipControls リクエストで他のユーザーにアクセス許可を付与するバケット ACL を持つバケットに対してバケット所有者の強制を設定すると、リクエストは失敗します。

Example ：既存のバケット ACL は公開読み取りアクセスを許可します

例えば、既存のバケット ACL がパブリック読み取りアクセスを許可している場合、これらの ACL アクセス許可をバケットポリシーに移行し、バケット ACL をデフォルトのプライベート ACL にリセッ

トするまで、オブジェクト所有権に[バケット所有者の強制] 設定を適用することはできません。詳細については、「ACL を無効にする前提条件」を参照してください。

次のバケット ACL の例は、公開読み取りアクセスを許可します。

```
{
    "Owner": {
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

次の例の put-bucket-ownership-controls AWS CLI コマンドは、オブジェクト所有権にバケット所有者の強制設定を適用します。

```
aws s3api put-bucket-ownership-controls --bucket DOC-EXAMPLE-BUCKET --ownership-
controls Rules=[{ObjectOwnership=BucketOwnerEnforced}]
```

バケット ACL は公開読み取りアクセスを許可するため、リクエストは失敗し、次のエラーコードを返します。

PutBucketOwnershipControls オペレーションを呼び出すときにエラーが発生しました。(InvalidBucketAclWithObjectOwnership): バケットに ObjectOwnership の BucketOwnerEnforced 設定で ACL を設定することはできません

Cross-Origin Resource Sharing (CORS) の使用

Cross-Origin Resource Sharing (CORS) は、特定のドメインにロードされたクライアントウェブアプリケーションが異なるドメイン内のリソースと通信する方法を定義します。Amazon S3 の CORS のサポートによって、Amazon S3 でリッチなクライアント側ウェブアプリケーションを構築し、Amazon S3 リソースへのクロスオリジンアクセスを選択的に許可できます。

このセクションでは、CORS の概要を示します。サブトピックでは、Amazon S3 コンソールを使用するか、Amazon S3 REST API や AWS SDK を使用して、プログラムによって CORS を有効にする方法について説明します。

Cross-Origin Resource Sharing: ユースケースのシナリオ

CORS のユースケースの例を以下に示します。

シナリオ 1

「Amazon S3 を使用して静的ウェブサイトをホスティングする」で説明されているように、`website` という名前の Amazon S3 バケットでウェブサイトをホストしているとします。ユーザーは、次のウェブサイトエンドポイントをロードします。

```
http://website.s3-website.us-east-1.amazonaws.com
```

このバケットに保存されているウェブページで JavaScript を使用し、バケットの Amazon S3 の API エンドポイント `website.s3.us-east-1.amazonaws.com` を使用して、同じバケットに対して認証済みの GET および PUT リクエストを行います。ブラウザは通常、それらのリクエストを許可しないように、JavaScript をブロックしますが、CORS を使用することにより、`website.s3-website.us-east-1.amazonaws.com` からのクロスオリジンリクエストを明示的に有効にするようにバケットを設定できます。

シナリオ 2

S3 バケットからウェブフォントをホストする必要があるとします。ここでも、ブラウザでウェブフォントを読み込むために CORS チェック (プリフライトチェックと呼ばれます) が必要です。ウェブフォントをホストしているバケットを、いずれのオリジンもこれらのリクエストを実行できるように設定します。

Amazon S3 でのバケットの CORS 設定の評価方法

Amazon S3 がブラウザからプリフライトリクエストを受け取ると、バケットの CORS 設定を評価し、受信ブラウザリクエストに一致する最初の CORSRule ルールを使用して、クロスオリジンリクエストを有効にします。ルールが一致するには、次の条件を満たしている必要があります。

- リクエストの Origin ヘッダーが AllowedOrigin エレメントに一致している必要があります。
- リクエストメソッド (GET や PUT など)、またはプリフライト Access-Control-Request-Method リクエストの場合は OPTIONS ヘッダーが、AllowedMethod エレメントのいずれかである必要があります。
- プリフライトリクエストのリクエストの Access-Control-Request-Headers ヘッダーにリストされているすべてのヘッダーが AllowedHeader エレメントに一致している必要があります。

> **Note**
>
> バケットの CORS 設定を有効にすると、ACL とポリシーが引き続き適用されます。

Object Lambda アクセスポイントで CORS をサポートする方法

S3 Object Lambda がブラウザからリクエストを受信するか、リクエストに Origin ヘッダーが含まれている場合、S3 Object Lambda は常に “AllowedOrigins":"*" ヘッダーフィールドを追加します。

CORS の使用の詳細については、次のトピックを参照してください。

トピック

- CORS の設定
- Cross-Origin Resource Sharing (CORS) の設定

CORS の設定

クロスオリジンリクエストを許可するようバケットを設定するには、CORS 設定を作成します。CORS 設定は、バケットへのアクセスを許可するオリジン、各オリジンでサポートされるオペレーション (HTTP メソッド)、その他のオペレーション固有情報を識別するルールを持つ XML ドキュメントです。設定には、最大 100 のルールを追加できます。CORS 設定を cors サブリソースとしてバケットに追加できます。

S3 コンソールで CORS を設定する場合は、JSON を使用して CORS 設定を作成する必要があります。新しい S3 コンソールでは、JSON CORS 設定のみがサポートされます。

CORS 設定とそのエレメントの詳細については、以下のトピックを参照してください。CORS 設定を追加する方法については、Cross-Origin Resource Sharing (CORS) の設定 を参照してください。

> **Important**
>
> S3 コンソールでは、CORS 設定は JSON である必要があります。

トピック

例 1

ウェブサイトへのアクセスに Amazon S3 ウェブサイトエンドポイントを使用する代わりに、所有するドメイン、例えば `example1.com` を使用してコンテンツを配信できます。独自のドメインの使用については、チュートリアル: Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定 を参照してください。

次の例の `cors` 設定には 3 つのルールがあり、`CORSRule` エレメントとして指定されています。

- 最初のルールは、`http://www.example1.com` オリジンからのクロスオリジン PUT、POST、および DELETE リクエストを許可します。このルールは、`Access-Control-Request-Headers` ヘッダーによって、プリフライト OPTIONS リクエスト内のすべてのヘッダーも許可します。プリフライト OPTIONS リクエストへのレスポンスとして、Amazon S3 はリクエストされたヘッダーを返します。
- 2 つ目のルールは最初のルールと同じクロスオリジンリクエストを許可しますが、このルールは別のオリジン `http://www.example2.com` に適用されます。

- 3 つ目のルールは、すべてのオリジンからのクロスオリジン GET リクエストを許可します。ワイルドカード文字 * は、すべてのオリジンを表します。

JSON

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example1.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example2.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": []
```

```
    }
]
```

XML

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example1.com</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>

   <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
   <AllowedOrigin>http://www.example2.com</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>

   <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
   <AllowedOrigin>*</AllowedOrigin>
   <AllowedMethod>GET</AllowedMethod>
 </CORSRule>
</CORSConfiguration>
```

例 2

次の CORS 設定に示すように、CORS 設定ではオプションの設定パラメータも使用できます。この例で、CORS 設定は `http://www.example.com` オリジンからのクロスオリジン PUT、POST、および DELETE の各リクエストを許可します。

JSON

```
[
    {
        "AllowedHeaders": [
            "*"
```

```
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example.com"
        ],
        "ExposeHeaders": [
            "x-amz-server-side-encryption",
            "x-amz-request-id",
            "x-amz-id-2"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

XML

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example.com</AllowedOrigin>
   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>
   <AllowedHeader>*</AllowedHeader>
  <MaxAgeSeconds>3000</MaxAgeSeconds>
  <ExposeHeader>x-amz-server-side-encryption</
ExposeHeader>
  <ExposeHeader>x-amz-request-id</
ExposeHeader>
  <ExposeHeader>x-amz-id-2</ExposeHeader>
 </CORSRule>
</CORSConfiguration>
```

前述の設定の CORSRule エレメントには、次のオプションのエレメントが含まれます。

- MaxAgeSeconds— 指定したリソースのプリフライト OPTIONS リクエストへの Amazon S3 レスポンスをブラウザでキャッシュする時間を秒単位で指定します (この例では 3,000)。レスポンスのキャッシュにより、元のリクエストが繰り返された場合に、ブラウザは Amazon S3 にプリフライトリクエストを送信する必要がありません。

- ExposeHeader – 顧客がアプリケーションから (例えば JavaScript x-amz-server-side-encryption オブジェクトから) アクセスできるレスポンスヘッダー (この例では x-amz-request-id、x-amz-id-2、および XMLHttpRequest) を識別します。

AllowedMethod エレメント

CORS 設定では、AllowedMethod エレメントに次の値を指定できます。

- GET
- PUT
- POST
- DELETE
- HEAD

AllowedOrigin エレメント

AllowedOrigin エレメントに、クロスドメインリクエストを許可するオリジンを指定します (http://www.example.com など)。オリジン文字列には、1 つのワイルドカード文字 (*) のみを含めることができます。例えば、http://*.example.com などです。オプションで、オリジンに * を指定して、すべてのオリジンでクロスオリジンリクエストを送信できるようにすることができます。さらに、https を指定して、セキュリティで保護されたオリジンのみを有効にすることもできます。

AllowedHeader エレメント

AllowedHeader エレメントは、Access-Control-Request-Headers ヘッダーによって、プリフライトリクエストで許可されるヘッダーを指定します。Access-Control-Request-Headers ヘッダー内の各ヘッダー名は、ルールの対応するエントリに一致する必要があります。Amazon S3 は、レスポンスで、リクエストされたヘッダーのうち許可されたヘッダーのみを送信します。Amazon S3 へのリクエストで使用できるヘッダーのサンプルリストについては、Amazon Simple Storage Service API リファレンスの一般的なリクエストヘッダー を参照してください。

ルール内の各 AllowedHeader 文字列には、最大 1 つのワイルドカード文字 (*) を含めることができます。例えば、<AllowedHeader>x-amz-*</AllowedHeader> は Amazon 固有のすべてのヘッダーを有効にします。

ExposeHeader エレメント

各 `ExposeHeader` エレメントは、顧客がアプリケーションから (例えば、JavaScript `XMLHttpRequest` オブジェクトから) アクセスできるようにするレスポンス内のヘッダーを識別します。一般的な Amazon S3 レスポンスヘッダーのリストについては、Amazon Simple Storage Service API リファレンスの一般的なリクエストヘッダー を参照してください。

MaxAgeSeconds エレメント

`MaxAgeSeconds` エレメントは、リソース、HTTP メソッド、およびオリジンによって識別されたプリフライトリクエストのレスポンスをブラウザでキャッシュできる時間を秒単位で指定します。

Cross-Origin Resource Sharing (CORS) の設定

Cross-Origin Resource Sharing (CORS) は、特定のドメインにロードされたクライアントウェブアプリケーションが異なるドメイン内のリソースと通信する方法を定義します。Amazon S3 の CORS のサポートによって、Amazon S3 でリッチなクライアント側ウェブアプリケーションを構築し、Amazon S3 リソースへのクロスオリジンアクセスを選択的に許可できます。

このセクションでは、Amazon S3 コンソール、Amazon S3 REST API、および AWS SDK を使用して CORS を有効にする方法について説明します。クロスオリジンリクエストを許可するようバケットを設定するには、CORS 設定をバケットに追加します。CORS 設定は、バケットへのアクセスを許可するオリジン、各オリジンでサポートされるオペレーション (HTTP メソッド)、およびその他のオペレーション固有情報を識別するルールを定義するドキュメントです。S3 コンソールでは、CORS 設定は JSON ドキュメントである必要があります。

JSON および XML での CORS 設定の例については、CORS の設定 を参照してください。

S3 コンソールの使用

このセクションでは、Amazon S3 コンソールを使用して Cross-Origin Resource Sharing (CORS) 設定を S3 バケットに追加する方法について説明します。

バケットで CORS を有効にすると、アクセスコントロールリスト (ACL) およびその他のアクセス許可ポリシーが引き続き適用されます。

> **Important**
>
> 新しい S3 コンソールでは、CORS 設定は JSON である必要があります。JSON および XML での CORS 設定の例については、CORS の設定 を参照してください。

CORS 設定を S3 バケットに追加するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets (バケット)] リストで、バケットポリシーを作成するバケットの名前を選択します。
3. [Permissions] を選択します。
4. [CORS (クロスオリジンリソース共有)] セクションで、[編集] を選択します。
5. [CORS configuration editor (CORS 構成エディタ)] テキストボックスに、新しい CORS 設定を入力またはコピーして貼り付けるか、既存の設定を編集します。

   CORS 設定は JSON ファイルです。エディタに入力するテキストは有効な JSON である必要があります。詳細については、「CORS の設定」を参照してください。
6. [Save changes] (変更の保存) をクリックします。

> **Note**
>
> Amazon S3 では、[CORS configuration editor (CORS 構成エディタ)] のタイトルの横に、バケットの Amazon リソースネーム (ARN) が表示されます。ARN の詳細については、Amazon Web Services 全般のリファレンス の「Amazon リソースネーム (ARN) および AWS サービスの名前空間」を参照してください。

AWS SDK の使用

AWS SDK を使用して、バケットの Cross-Origin Resource Sharing (CORS) を管理できます。CORS の詳細については、Cross-Origin Resource Sharing (CORS) の使用 を参照してください。

以下の例を参照してください。

- CORS 設定を作成し、バケット上で設定を指定する
- 設定を取得し、ルールを追加してその設定を変更する
- 変更された設定をバケットに追加する
- 設定を削除する

Java

Example

Example

作業サンプルの作成方法およびテスト方法については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketCrossOriginConfiguration;
import com.amazonaws.services.s3.model.CORSRule;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CORS {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        // Create two CORS rules.
        List<CORSRule.AllowedMethods> rule1AM = new
 ArrayList<CORSRule.AllowedMethods>();
        rule1AM.add(CORSRule.AllowedMethods.PUT);
        rule1AM.add(CORSRule.AllowedMethods.POST);
        rule1AM.add(CORSRule.AllowedMethods.DELETE);
        CORSRule rule1 = new
 CORSRule().withId("CORSRule1").withAllowedMethods(rule1AM)
                .withAllowedOrigins(Arrays.asList("http://*.example.com"));

        List<CORSRule.AllowedMethods> rule2AM = new
 ArrayList<CORSRule.AllowedMethods>();
        rule2AM.add(CORSRule.AllowedMethods.GET);
```

```
        CORSRule rule2 = new
CORSRule().withId("CORSRule2").withAllowedMethods(rule2AM)
                .withAllowedOrigins(Arrays.asList("*")).withMaxAgeSeconds(3000)
                .withExposedHeaders(Arrays.asList("x-amz-server-side-encryption"));

        List<CORSRule> rules = new ArrayList<CORSRule>();
        rules.add(rule1);
        rules.add(rule2);

        // Add the rules to a new CORS configuration.
        BucketCrossOriginConfiguration configuration = new
BucketCrossOriginConfiguration();
        configuration.setRules(rules);

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Add the configuration to the bucket.
            s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

            // Retrieve and display the configuration.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            printCORSConfiguration(configuration);

            // Add another new rule.
            List<CORSRule.AllowedMethods> rule3AM = new
ArrayList<CORSRule.AllowedMethods>();
            rule3AM.add(CORSRule.AllowedMethods.HEAD);
            CORSRule rule3 = new
CORSRule().withId("CORSRule3").withAllowedMethods(rule3AM)
                    .withAllowedOrigins(Arrays.asList("http://www.example.com"));

            rules = configuration.getRules();
            rules.add(rule3);
            configuration.setRules(rules);
            s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

            // Verify that the new rule was added by checking the number of rules in
the
            // configuration.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
```

```
            System.out.println("Expected # of rules = 3, found " +
 configuration.getRules().size());

            // Delete the configuration.
            s3Client.deleteBucketCrossOriginConfiguration(bucketName);
            System.out.println("Removed CORS configuration.");

            // Retrieve and display the configuration to verify that it was
            // successfully deleted.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            printCORSConfiguration(configuration);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    private static void printCORSConfiguration(BucketCrossOriginConfiguration
 configuration) {
        if (configuration == null) {
            System.out.println("Configuration is null.");
        } else {
            System.out.println("Configuration has " +
 configuration.getRules().size() + " rules\n");

            for (CORSRule rule : configuration.getRules()) {
                System.out.println("Rule ID: " + rule.getId());
                System.out.println("MaxAgeSeconds: " + rule.getMaxAgeSeconds());
                System.out.println("AllowedMethod: " + rule.getAllowedMethods());
                System.out.println("AllowedOrigins: " + rule.getAllowedOrigins());
                System.out.println("AllowedHeaders: " + rule.getAllowedHeaders());
                System.out.println("ExposeHeader: " + rule.getExposedHeaders());
                System.out.println();
            }
        }
    }
}
```

.NET

Example

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CORSTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CORSConfigTestAsync().Wait();
        }
        private static async Task CORSConfigTestAsync()
        {
            try
            {
                // Create a new configuration request and add two rules
                CORSConfiguration configuration = new CORSConfiguration
                {
                    Rules = new System.Collections.Generic.List<CORSRule>
                        {
                          new CORSRule
                          {
                            Id = "CORSRule1",
                            AllowedMethods = new List<string> {"PUT", "POST",
 "DELETE"},
```

```
                            AllowedOrigins = new List<string> {"http://
*.example.com"}
                        },
                        new CORSRule
                        {
                          Id = "CORSRule2",
                          AllowedMethods = new List<string> {"GET"},
                          AllowedOrigins = new List<string> {"*"},
                          MaxAgeSeconds = 3000,
                          ExposeHeaders = new List<string> {"x-amz-server-side-
encryption"}
                        }
                      }
                };

                // Add the configuration to the bucket.
                await PutCORSConfigurationAsync(configuration);

                // Retrieve an existing configuration.
                configuration = await RetrieveCORSConfigurationAsync();

                // Add a new rule.
                configuration.Rules.Add(new CORSRule
                {
                    Id = "CORSRule3",
                    AllowedMethods = new List<string> { "HEAD" },
                    AllowedOrigins = new List<string> { "http://www.example.com" }
                });

                // Add the configuration to the bucket.
                await PutCORSConfigurationAsync(configuration);

                // Verify that there are now three rules.
                configuration = await RetrieveCORSConfigurationAsync();
                Console.WriteLine();
                Console.WriteLine("Expected # of rulest=3; found:{0}",
 configuration.Rules.Count);
                Console.WriteLine();
                Console.WriteLine("Pause before configuration delete. To continue,
 click Enter...");
                Console.ReadKey();

                // Delete the configuration.
                await DeleteCORSConfigurationAsync();
```

```
                // Retrieve a nonexistent configuration.
                configuration = await RetrieveCORSConfigurationAsync();
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        static async Task PutCORSConfigurationAsync(CORSConfiguration configuration)
        {

            PutCORSConfigurationRequest request = new PutCORSConfigurationRequest
            {
                BucketName = bucketName,
                Configuration = configuration
            };

            var response = await s3Client.PutCORSConfigurationAsync(request);
        }

        static async Task<CORSConfiguration> RetrieveCORSConfigurationAsync()
        {
            GetCORSConfigurationRequest request = new GetCORSConfigurationRequest
            {
                BucketName = bucketName

            };
            var response = await s3Client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
            return configuration;
        }

        static async Task DeleteCORSConfigurationAsync()
        {
```

```
            DeleteCORSConfigurationRequest request = new
 DeleteCORSConfigurationRequest
            {
                BucketName = bucketName
            };
            await s3Client.DeleteCORSConfigurationAsync(request);
        }

        static void PrintCORSRules(CORSConfiguration configuration)
        {
            Console.WriteLine();

            if (configuration == null)
            {
                Console.WriteLine("\nConfiguration is null");
                return;
            }

            Console.WriteLine("Configuration has {0} rules:",
 configuration.Rules.Count);
            foreach (CORSRule rule in configuration.Rules)
            {
                Console.WriteLine("Rule ID: {0}", rule.Id);
                Console.WriteLine("MaxAgeSeconds: {0}", rule.MaxAgeSeconds);
                Console.WriteLine("AllowedMethod: {0}", string.Join(", ",
 rule.AllowedMethods.ToArray()));
                Console.WriteLine("AllowedOrigins: {0}", string.Join(", ",
 rule.AllowedOrigins.ToArray()));
                Console.WriteLine("AllowedHeaders: {0}", string.Join(", ",
 rule.AllowedHeaders.ToArray()));
                Console.WriteLine("ExposeHeader: {0}", string.Join(", ",
 rule.ExposeHeaders.ToArray()));
            }
        }
    }
}
```

REST API の使用

バケットで CORS 設定を指定するには、AWS Management Consoleを使用できます。必要に応じて、REST リクエストを直接送信することもできます。Amazon Simple Storage Service API リファ

レンスの以下のセクションでは、CORS 設定に関連する REST API アクションについて説明しています。

- PutBucketCors
- GetBucketCors
- DeleteBucketCors
- OPTIONS オブジェクト

Amazon S3 でのログ記録とモニタリング

モニタリングは、Amazon S3 および AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできるように、AWS ソリューションのすべての部分からモニタリングデータを収集する必要があります。AWS には、Amazon S3 リソースをモニタリングし、潜在的なインシデントに対応するための複数のツールが用意されています。

詳細については、「Amazon S3 のモニタリング」を参照してください。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

Amazon CloudWatch アラーム

Amazon CloudWatch アラームを使用して、指定した期間中、1 つのメトリクスをモニタリングします。メトリクスが特定の閾値を超えると、Amazon SNS トピックまたは AWS Auto Scaling ポリシーに通知が送信されます。CloudWatch アラームは、特定の状態にあるという理由ではアクションを呼び出しません。状態が変わり、それが指定した期間だけ維持される必要があります。詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。

AWS CloudTrail ログ

CloudTrail は、Amazon S3 のユーザー、ロール、または AWS のサービスによって実行されたアクションの記録を提供します。CloudTrail で収集された情報を使用して、Amazon S3 に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。詳細については、「AWS CloudTrail を使用した Amazon S3 API コールのログ記録」を参照してください。

Amazon GuardDuty

Amazon GuardDuty は、アカウント、コンテナ、ワークロード、および AWS 環境内のデータを継続的にモニタリングし、S3 バケットに対する潜在的な脅威やセキュリティリスクを特定する脅威検出サービスです。また、GuardDuty は、検出した脅威に関する詳細なコンテキストも提供します。GuardDuty は、脅威の AWS CloudTrail 管理ログを監視し、セキュリティ関連情報を表示

します。例えば、GuardDuty には、リクエストを行ったユーザー、リクエストが行われた場所、リクエストされた具体的な API など、環境で異常である可能性がある API リクエストの要素が含まれます。GuardDuty S3 Protection は、CloudTrail によって収集された S3 データイベントを監視し、環境内のすべての S3 バケットで異常および悪意のある可能性のある動作を特定します。

Amazon S3 アクセスログ

サーバーアクセスログでは、バケットに対して行われたリクエストの詳細なレコードが提供されます。サーバーアクセスのログは、多くのアプリケーションに役立ちます。例えば、アクセスのログ情報は、セキュリティやアクセスの監査に役立ちます。詳細については、「サーバーアクセスログによるリクエストのログ記録」を参照してください。

AWS Trusted Advisor

Trusted Advisor は、AWS の数十万のお客様にサービスを提供することにより得られた、運用実績から学んだベストプラクティスを活用しています。Trusted Advisor はお客様の AWS 環境を検査し、システムの可用性とパフォーマンスを向上させたりセキュリティギャップを埋めたりする機会がある場合には、推奨事項を作成します。すべての AWS のお客様は、Trusted Advisor の 5 つのチェックにアクセスできます。ビジネスまたはエンタープライズサポートプランをご利用のお客様は、すべての Trusted Advisor チェックを表示できます。

Trusted Advisor には、以下の Amazon S3 関連のチェックがあります。

- Amazon S3 バケットのログ記録設定のチェック。
- オープンなアクセス許可がある Amazon S3 バケットのセキュリティチェック。
- バージョニングが有効になっていない、またはバージョニングが停止されている Amazon S3 バケットの耐障害性チェック。

詳細については、AWS Support ユーザーガイドの AWS Trusted Advisor を参照してください。

以下のセキュリティのベストプラクティスも口グ記録とモニタリングに対処します。

- Identify and audit all your Amazon S3 buckets
- Implement monitoring using Amazon Web Services monitoring tools
- AWS Config の有効化
- Enable Amazon S3 server access logging
- Use CloudTrail
- Monitor Amazon Web Services security advisories

Amazon S3 のコンプライアンス検証

Amazon S3 のセキュリティおよびコンプライアンスは、以下を含む複数の AWS コンプライアンスプログラムの一環として、サードパーティーの監査者により評価されます。

- System and Organization Controls (SOC)
- Payment Card Industry Data Security Standard (PCI DSS)
- Federal Risk and Authorization Management Program (FedRAMP)
- Health Insurance Portability and Accountability Act (HIPAA)

AWS は、コンプライアンスプログラムのターゲット範囲内の AWS サービスで、特定のコンプライアンスプログラムのターゲット範囲内における AWS サービス一覧を頻繁に更新しています。

サードパーティーの監査レポートは、AWS Artifact を使用してダウンロードできます。詳細については、AWS Artifact のレポートのダウンロード を参照してください。

AWS コンプライアンスプログラムの詳細については、AWS コンプライアンスプログラム を参照してください。

Amazon S3 を使用する際のお客様のコンプライアンス責任は、組織のデータの機密性や組織のコンプライアンス目的、適用可能な法律、規制によって決定されます。Amazon S3 の使用が HIPAA、PCI、FedRAMP などの標準に準拠していることを前提としている場合、AWS は以下を支援するリソースを提供します。

- セキュリティおよびコンプライアンスのクイックスタートガイドでは、AWS のデプロイメントセキュリティやコンプライアンスに重点を置いたベースライン環境におけるアーキテクチャ上の考慮事項や手順について説明しています。
- HIPAA セキュリティおよびコンプライアンス向けアーキテクチャ設計では、企業が AWS を使用して HIPAA 要件を満たす方法について説明します。
- AWS コンプライアンスリソースを使用すると、業界や地域で使用できるワークブックとガイドとを選択できます。
- AWS Config を使用すると、社内プラクティス、業界ガイドライン、およびに対するリソースの設定の準拠状態を評価できます。
- AWS Security Hub を使用すると、AWS 内のセキュリティ状態を包括的に表示し、セキュリティ業界の標準およびベストプラクティスへの準拠を確認できます。

- S3 オブジェクトロックの使用を使用すると、特定の種類の帳簿および記録情報に対して、Write Once Read Many (WORM) データストレージの使用を必須とする、金融サービス規制機関 (SEC、FINRA、CFTC など) の技術的要件を満たすことができます。
- Amazon S3 インベントリ は、ビジネス、コンプライアンス、規制上のニーズに対応して、オブジェクトのレプリケーションと暗号化のステータスを監査し、レポートするのに役立ちます。

Amazon S3 の耐障害性

AWS グローバルインフラストラクチャは、リージョンおよびアベイラビリティーゾーンを中心に構築されています。AWS リージョンには、低レイテンシー、高スループット、高冗長性のネットワークで接続されている複数の物理的に独立、分離されたアベイラビリティーゾーンがあります。これらのアベイラビリティーゾーンを利用すると、アプリケーションとデータベースを効率的に設計して運用できます。アベイラビリティーゾーンは、従来の単一データセンターのインフラストラクチャや複数データセンターのインフラストラクチャよりも可用性、耐障害性、および拡張性が優れています。特に地理的に離れた場所間でデータをレプリケートする必要がある場合は、オブジェクトのレプリケーションを使用できます。これにより、さまざまな AWS リージョンにあるバケット間でオブジェクトを自動的に非同期コピーできます。

各 AWS リージョンには、複数のアベイラビリティーゾーンがあります。耐障害性と低レイテンシーのために、同じリージョン内の複数のアベイラビリティーゾーンにアプリケーションをデプロイできます。アベイラビリティーゾーンは高速なプライベート光ファイバーネットワーキングで相互に接続されているため、アプリケーションがアベイラビリティーゾーン間で中断なく自動的にフェイルオーバーできるようなアーキテクチャを簡単に設計できます。

AWS リージョン とアベイラビリティーゾーンの詳細については、AWS グローバルインフラストラクチャ を参照してください。

Amazon S3 では、AWS グローバルインフラストラクチャに加えて、データの耐障害性とバックアップのニーズに対応できるように複数の機能を提供しています。

ライフサイクル設定

ライフサイクル設定は、Amazon S3 がオブジェクトのグループに適用するアクションを定義するルールのセットです。ライフサイクル設定ルールを使用すると、オブジェクトのより安価なストレージクラスへの移行、アーカイブ、削除を Amazon S3 に指定できます。詳細については、「ストレージのライフサイクルの管理」を参照してください。

バージョニング

バージョニングとは、同じバケット内でオブジェクトの複数のバリアントを保持する手段です。バージョニングを使用して、Amazon S3 バケットに格納されたあらゆるオブジェクトのあらゆるバージョンを、格納、取得、復元することができます。バージョニングを使用すれば、意図しないユーザーアクションからもアプリケーション障害からも、簡単に復旧できます。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

S3 オブジェクトロック

S3 オブジェクトロックでは、Write Once Read Many (WORM) モデルを使用してオブジェクトを保存できます。S3 オブジェクトロックを使用すると、オブジェクトが固定期間または無期限に削除または上書きされるのを防止できます。S3 オブジェクトロックを使用して、WORM ストレージを必要とする規制要件を満たしたり、オブジェクトの変更や削除に対する保護レイヤーを追加したりできます。詳細については、「S3 オブジェクトロックの使用」を参照してください。

ストレージクラス

Amazon S3 では、ワークロードの要件に応じて、幅広いストレージクラスが提供されています。S3 標準 — IA と S3 1 ゾーン – IA ストレージクラスは、月に約 1 回アクセスし、ミリ秒単位のアクセスが必要になるデータ用に設計されています。S3 Glacier インスタント検索ストレージクラスは、四半期に約 1 回アクセスするミリ秒のアクセスでアクセスされる長期間有効なアーカイブデータ用に設計されています。バックアップなど、即時アクセスを必要としないアーカイブデータについては、S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスを使用できます。詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。

以下のセキュリティのベストプラクティスも耐障害性に対処します。

- Enable versioning
- Consider Amazon S3 cross-region replication
- Identify and audit all your Amazon S3 buckets

Amazon S3 バックアップの暗号化

Amazon S3 を使用してバックアップを保存する場合、バックアップの暗号化は各バケットの設定によって異なります。Amazon S3 により、S3 バケットのデフォルト暗号化の動作を設定できます。バケットにデフォルト暗号化を設定して、バケットに保存される際すべてのオブジェクトが暗号化されるようにします。デフォルト暗号化は、AWS KMS に保存されたキー (SSE-KMS) をサポートしています。詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。

バージョニングとオブジェクトロックの詳細については、以下のトピックを参照してください。S3 バケットでのバージョニングの使用S3 オブジェクトロックの使用

Amazon S3 のインフラストラクチャセキュリティ

マネージドサービスとして、Amazon S3 は AWS のセキュリティの柱で説明されたグローバルネットワークセキュリティ手順 AWSWell-Architected フレームワークによって保護されます。

ネットワークを経由した Amazon S3 へのアクセスは、AWS が発行する API を介して行われます。クライアントは Transport Layer Security (TLS) 1.2 をサポートしている必要があります。TLS 1.3 もサポートすることをお勧めします (この推奨事項の詳細については、AWS セキュリティブログの「TLS 1.3 による AWS クラウド接続の高速化」を参照してください)。また、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもサポートしている必要があります。さらに、AWS Signature V4 または AWS Signature V2 を使用してリクエストに署名する必要があります。そのためには、有効な認証情報を提供する必要があります。

これらの API はネットワークの任意の場所から呼び出すことができます。ただし、Amazon S3 はリソースベースのアクセスポリシーをサポートしており、それらのポリシーには、ソース IP アドレスに基づく制限を含めることができます。Amazon S3 のバケットポリシーを使用して、特定の Virtual Private Cloud (VPC) エンドポイントまたは特定の VPC からのバケットへのアクセスを管理できます。これにより、実質的に AWS ネットワーク内の特定の VPC からのみ特定の Amazon S3 バケットへのネットワークアクセスが分離されます。詳細については、「バケットポリシーを使用した VPC エンドポイントからのアクセスコントロール」を参照してください。

以下のセキュリティのベストプラクティスも Amazon S3 でのインフラストラクチャのセキュリティに対処します。

- Consider VPC endpoints for Amazon S3 access
- Identify and audit all your Amazon S3 buckets

Amazon S3 での設定と脆弱性の分析

AWS は、ゲストオペレーティングシステム (OS) やデータベースへのパッチ適用、ファイアウォール設定、災害対策などの基本的なセキュリティタスクを処理します。これらの手順は適切な第三者によって確認され、証明されています。詳細については、以下のリソース を参照してください。

- Amazon S3 のコンプライアンス検証
- 責任共有モデル
- Amazon Web Services：セキュリティプロセスの概要

以下のセキュリティのベストプラクティスも Amazon S3 での設定と脆弱性の分析に対処します。

- Identify and audit all your Amazon S3 buckets
- AWS Config の有効化

Amazon S3 のセキュリティのベストプラクティス

Amazon S3 には、独自のセキュリティポリシーを策定および実装する際に考慮すべきさまざまなセキュリティ機能が用意されています。以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションを提供するものではありません。これらのベストプラクティスは顧客の環境に必ずしも適切または十分でない可能性があるので、処方箋ではなく、あくまで有用な推奨事項とお考えください。

トピック

Amazon S3 のセキュリティベストプラクティス

以下は、セキュリティ問題の防止に役立つ Amazon S3 でのベストプラクティスです。

アクセスコントロールリスト (ACL) の無効化

S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、バケットにアップロードされたオブジェクトの所有権を制御し、ACL を無効または有効にするのに使用できます。デフォルトでは、オブジェクト所有権は [バケット所有者の強制] により設定され、すべての ACL は無効になっています。ACL が無効になっている場合、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。

Amazon S3 の最近のユースケースの大部分ではアクセスコントロールリスト (ACL) を使用する必要がなくなりました。オブジェクトごとに個別に制御する必要があるまれな状況を除き、ACL を無効にすることをお勧めします。バケットのすべてのオブジェクトの ACL を無効にして、所有権を取得するには、S3 オブジェクト所有権のバケット所有者の強制設定を適用します。ACL を無効にすると、別の AWS アカウント によってアップロードされたオブジェクトを含むバケットを簡単に維持できます。

ACL が無効になっている場合、データのアクセスコントロールは次のようなポリシーに基づきます。

- AWS Identity and Access Management IAM ユーザーポリシー
- S3 バケットポリシー
- 仮想プライベートクラウド (VPC) エンドポイントポリシー

- AWS Organizations サービスコントロールポリシー (SCP)

ACL の無効化により、アクセス許可の管理と監査が簡素化されます。デフォルトでは、ACL は無効になっています。既存のバケットに対して ACL を無効にすることもできます。既存のバケットに既にオブジェクトが含まれている場合、ACL を無効にすると、オブジェクトとバケット ACL はアクセス評価プロセスの一部ではなくなります。この場合、アクセスはポリシーに基づいて許可または拒否されます。

ACL を無効にする前に、次のことを確認してください。

- バケットポリシーを確認して、アカウント外のバケットへのアクセス権を付与するすべての方法をカバーすることを確認します。
- バケット ACL をデフォルト (バケット所有者にフルコントロール) にリセットします。

ACL を無効にすると、以下の動作が発生します。

- バケットは、ACL を指定しない PUT リクエスト、またはバケット所有者のフルコントロール ACL を含む PUT リクエストのみを受け付けます。これらの ACL には、`bucket-owner-full-control` 既定 ACL または XML で表現された ACL と同等の形式が含まれます。
- バケット所有者の完全制御 ACL をサポートする既存のアプリケーションには影響はありません。
- 他の ACL (例えば、特定 AWS アカウント のへのカスタム許可) を含む PUT 要求は失敗し、`AccessControlListNotSupported` エラーコードとともに HTTPHTTP ステータスコード `400 (Bad Request)` を返します。

詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

Amazon S3 バケットに正しいポリシーが使用され、バケットが公開されていないことを確認する

インターネット上のだれもが S3 バケットを読み書きできるように明示的にリクエストしない限り、S3 バケットが非公開であることを確認する必要があります。以下に示しているのは、パブリックアクセスをブロックするために実行できるいくつかのステップです。

- S3 パブリックアクセスのブロックを使用します。S3 パブリックアクセスブロックでは、Amazon S3 リソースへのパブリックアクセスを制限する一元管理を簡単に設定することができます。これらの一元管理は、リソースの作成方法に関係なく適用されます。詳細については、「Amazon S3 ストレージへのパブリックアクセスのブロック」を参照してください。

- `"Principal": "*"` (事実上「誰でも」という意味) などのワイルドカード ID を許可する Amazon S3 バケットポリシーを特定します。また、ワイルドカードアクション `"*"` (ユーザーが Amazon S3 バケットで任意のアクションを実行できるようにする) ポリシーも探します。
- 同様に、「全員」または「任意の認証済み AWS ユーザー」への読み取り、書き込み、またはフルアクセスを許可する Amazon S3 バケットのアクセスコントロールリスト (ACL) を探します。
- `ListBuckets` API オペレーションを使用して、すべての Amazon S3 バケットをスキャンします。次に、`GetBucketAcl`、`GetBucketWebsite`、`GetBucketPolicy` を使用して、各バケットがアクセスコントロールと設定がコンプライアンス要件を満たしているかどうかを判断します。
- AWS Trusted Advisor を使用して、Amazon S3 の実装を検査します。
- s3-bucket-public-read-prohibited および s3-bucket-public-write-prohibited マネージド AWS Config ルール を使用して、継続的な検出コントロールを実施することを検討します。

詳細については、「Amazon S3 用 Identity and Access Management」を参照してください。

Amazon GuardDuty を使用して Amazon S3 バケットに対する潜在的な脅威を特定する

Amazon GuardDuty は、AWS 環境内のアカウント、コンテナ、ワークロード、データに対する潜在的な脅威を特定する脅威検知サービスです。Amazon GuardDuty は、機械学習 (ML) モデルと異常および脅威検出機能を使用して、さまざまなデータソースを継続的に監視し、環境内の潜在的なセキュリティリスクと悪意のあるアクティビティを特定して優先順位を付けます。GuardDuty を有効にすると、AWS CloudTrail 管理イベント、VPC フローログ、DNS ログを含む基盤データソースの脅威検出が提供されます。脅威検出を S3 バケット内のデータプレーンイベントに拡張するには、GuardDuty S3 Protection 機能を有効にします。この機能は、データ流出や Tor ノードを介した S3 バケットへの疑わしいアクセスなどの脅威を検出します。また GuardDuty は、環境での通常のベースラインパターンを確立し、潜在的な異常な動作を特定すると、侵害された可能性のある S3 バケットまたは AWS 認証情報の修復に役立つコンテキスト情報も提供します。詳細については、「GuardDuty」を参照してください。

最小特権アクセスの実装

アクセス許可を付与する場合、どのユーザーにどの Amazon S3 リソースに対するアクセス許可を付与するかは、ユーザーが決定します。つまり、該当リソースに対して許可する特定のアクションを有効にするということです。このため、タスクを実行するために必要な許可のみを付与することをお勧めします。最小限の特権アクセスの実装は、セキュリティリスクはもちろん、エラーや悪意ある行動によってもたらされる可能性のある影響を減らす上での基本となります。

以下のツールは、最小限の特権アクセスを実装するために使用できます。

- Amazon S3 のポリシーアクション および IAM エンティティのアクセス許可境界
- Amazon S3 での IAM の機能
- アクセスコントロールリスト (ACL) の概要
- サービスコントロールポリシー

上記のメカニズムを採用する場合に何を考慮すべきかに関するガイダンスについては、Amazon S3 用 Identity and Access Management を参照してください。

Amazon S3 アクセスを必要とするアプリケーションと AWS のサービス に IAM ロールを使用する

Amazon EC2 または他の AWS のサービス のアプリケーションが Amazon S3 リソースにアクセスするためには、AWS API リクエストに有効な AWS 認証情報が含まれている必要があります。AWS 認証情報を、アプリケーションまたは Amazon EC2 インスタンスに直接保存しないことをお勧めします。これらは自動的にローテーションされない長期的な認証情報であり、漏洩するとビジネスに大きな影響が及ぶ場合があります。

代わりに、IAM ロールを使用して、Amazon S3 にアクセスする必要があるアプリケーションまたはサービスの一時的な認証情報を管理します。ロールを使用する場合は、Amazon EC2 インスタンスまたは AWS のサービス (AWS Lambda など) に長期の認証情報 (ユーザー名とパスワード、アクセスキーなど) を配布する必要はありません。ロールは、アプリケーションが他の AWS リソースの呼び出しを行うときに使用できる一時的な許可を付与します。

詳細については、IAM ユーザーガイドにある下記のトピックを参照してください。

- IAM ロール
- ロールの一般的なシナリオ: ユーザー、アプリケーション、およびサービス

保管時のデータ暗号化の検討

Amazon S3 で保管時のデータを保護するには、次のようなオプションがあります。

- サーバー側の暗号化 – すべての Amazon S3 バケットにはデフォルトで暗号化が設定されており、S3 バケットにアップロードされたすべての新しいオブジェクトは保存時に自動的に暗号化されます。Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) は、Amazon S3 のすべてのバケットでのデフォルトの暗号化設定です。別のタイプの暗号化を使用するには、S3 PUT リクエストで使用するサーバー側の暗号化のタイプを指定するか、宛先バケットにデフォルトの暗号化設定を設定できます。

  Amazon S3 には、次のサーバー側の暗号化オプションもあります。

  - AWS Key Management Service (AWS KMS) キー (SSE-KMS) によるサーバー側の暗号化

- AWS Key Management Service (AWS KMS) キーによる二層式サーバー側の暗号化 (DSSE-KMS)
- 顧客提供のキーを用いたサーバー側の暗号化 (SSE-C)。

詳細については、「サーバー側の暗号化によるデータの保護」を参照してください。

- クライアント側の暗号化 – クライアント側でデータを暗号化し、暗号化したデータを Amazon S3 にアップロードします。この場合、暗号化プロセス、暗号化キー、関連ツールはお客様が管理してください。サーバー側の暗号化と同様に、クライアント側の暗号化は、データ自体を保存するメカニズムとは異なるメカニズムで保存されているキーを使用してデータを暗号化することで、リスクを軽減するのに役立ちます。

  Amazon S3 は複数のクライアント側の暗号化オプションを提供します。詳細については、「クライアント側の暗号化を使用したデータの保護」を参照してください。

送信時のデータの暗号化を強制する

HTTPS (TLS) を使用すると、潜在的な攻撃者が中間者攻撃または同様の攻撃を使用してネットワークトラフィックを盗聴または操作することを防止できます。Amazon S3 バケットのポリシーで aws:SecureTransport 条件を使用して、HTTPS (TLS) 経由での暗号化された接続のみを許可することをお勧めします。

> **Important**
>
> AWS はパブリックに信頼された証明書の固定をサポートしていないため、アプリケーションでは Amazon S3 TLS 証明書を固定しないことをお勧めします。S3 は証明書を自動的に更新し、更新は証明書の有効期限が切れる前に行われます。証明書の更新では、新しいパブリックキーとプライベートキーのペアが生成されます。新しいパブリックキーで最近更新された S3 証明書を固定する場合、アプリケーションが新しい証明書を使用するまで S3 に接続できません。

また、s3-bucket-ssl-requests-only マネージド AWS Config ルールを使用した継続的な検出コントロールの実装を検討してください。

S3 オブジェクトロックの検討

S3 オブジェクトロックでは、Write Once Read Many (WORM) モデルを使用してオブジェクトを保存できます。S3 オブジェクトロックは、データの誤った削除や不適切な削除を防ぐのに役立ちます。例えば、AWS CloudTrail ログを保護するために S3 オブジェクトロックを使用できます。

詳細については、「S3 オブジェクトロックの使用」を参照してください。

S3 バージョニングの有効化

S3 バージョニングとは、同じバケット内でオブジェクトの複数のバリアントを保持する手段です。バージョニングを使用して、 バケットに保存されたあらゆるオブジェクトのあらゆるバージョンを保存、取得、復元することができます。バージョニングを使用すれば、意図しないユーザーアクションからもアプリケーション障害からも、簡単に復旧できます。

また、s3-bucket-versioning-enabled マネージド AWS Config ルールを使用した継続的な検出コントロールの実装を検討してください。

詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

S3 クロスリージョンレプリケーションの検討

Amazon S3 はデフォルトで地理的に異なる複数のアベイラビリティーゾーンにデータを保存しますが、コンプライアンス要件によっては、さらに離れた場所にデータを保存することが要求される場合があります。S3 クロスリージョンレプリケーション (CRR) を使うと、データを遠く離れた AWS リージョン にレプリケートできるため、これらの要件を満たすのに役立ちます。CRR では、異なる AWS リージョンのバケット間でオブジェクトを自動的に非同期コピーできます。詳細については、「オブジェクトのレプリケーション」を参照してください。

> **Note**
>
> CRR では、ソースとターゲットの両方の S3 バケットでバージョニングが有効になっている必要があります。

また、s3-bucket-replication-enabled マネージド AWS Config ルールを使用した継続的な検出コントロールの実装を検討してください。

Amazon S3 アクセス用の VPC エンドポイントの検討

Amazon S3 の 仮想プライベートクラウド (VPC) エンドポイントは、Amazon S3 への接続のみを許可する VPC 内の論理エンティティです。VPC エンドポイントは、トラフィックがオープンインターネットを通過するのを防ぐのに役立ちます。

Amazon S3 の VPC エンドポイントは、Amazon S3 データへのアクセスを制御するいくつかの方法を提供します。

- S3 バケットポリシーを使用して、特定の VPC エンドポイントを通じて許可されるリクエスト、ユーザー、またはグループを管理できます。

- S3 バケットポリシーを使用して、S3 バケットへのアクセスが可能な VPC または VPC エンドポイントをコントロールできます。
- インターネットゲートウェイのない VPC を使用すると、データの流出を防ぐのに役立ちます。

詳細については、「バケットポリシーを使用した VPC エンドポイントからのアクセスコントロール」を参照してください。

マネージド AWS セキュリティサービスを使用したデータセキュリティの監視

いくつかのマネージド AWS セキュリティサービスは、Amazon S3 データのセキュリティとコンプライアンスのリスクを特定、評価、監視するのに役立ちます。これらのサービスは、こうしたリスクからデータを保護するのにも役立ちます。これらのサービスには、単一の AWS アカウント Amazon S3 リソースから数千のアカウントにまたがる組織向けのリソースに拡張できるように設計された、自動検知、モニタリング、保護機能が含まれます。

詳細については、「マネージド AWS セキュリティサービスによるデータセキュリティのモニタリング」を参照してください。

Amazon S3 のモニタリングと監査のベストプラクティス

以下は、潜在的なセキュリティ上の弱点とインシデントを検出するために役立つ Amazon S3 でのベストプラクティスです。

すべての Amazon S3 バケットを特定して監査

IT アセットの特定はガバナンスとセキュリティの重要な側面です。セキュリティの状態を評価し、潜在的な弱点に対処するには、すべての Amazon S3 リソースが見えていなければなりません。リソースを監査するには、以下を実行することをお勧めします。

- タグエディターを使用してセキュリティまたは監査で注意を要するリソースを識別してタグ付けするには、これらのリソースを検索する必要があるときにそれらのタグを使用します。詳細については、「AWS リソースのタグ付けユーザーガイド」の「タグ付けするリソースの検索」を参照してください。
- S3 インベントリを使用して、ビジネス、コンプライアンス、および規制上のニーズに応じて、オブジェクトのレプリケーションと暗号化のステータスを監査し、レポートします。詳細については、「Amazon S3 インベントリ」を参照してください。
- Amazon S3 リソースのリソースグループを作成します。詳細については、「AWS Resource Groups ユーザーガイド」の「 リソースグループとは」を参照してください。

AWS モニタリングツールによるモニタリングの実装

モニタリングは、Amazon S3 および AWS ソリューションの信頼性、セキュリティ、可用性、パフォーマンスを維持する上で重要なエレメントです。AWS では、Amazon S3 およびその他の AWS のサービス をモニタリングするのに役立つツールとサービスを提供しています。例えば、Amazon S3 の Amazon CloudWatch メトリクス (特に、`PutRequests`、`GetRequests`、`4xxErrors`、`DeleteRequests`) をモニタリングできます。詳細については、Amazon CloudWatch によるメトリクスのモニタリングおよびAmazon S3 のモニタリングを参照してください。

別の例については、例: Amazon S3 バケットのアクティビティ を参照してください。この例では、バケットのポリシー、バケットのライフサイクル、またはバケットのレプリケーション設定の PUT または `DELETE` に対して、あるいはバケット ACL の PUT に対して Amazon S3 API コールが行われたときにトリガーされる CloudWatch アラームを作成する方法について説明します。

Amazon S3 サーバーアクセスログを有効にする

サーバーアクセスのログには、バケットに対するリクエストの詳細が記録されます。サーバーアクセスログは、セキュリティやアクセス監査の参考になることがあり、顧客基盤について知り、Amazon S3 の請求を理解するのに役立ちます。サーバーアクセスログ記録を有効にする手順については、サーバーアクセスログによるリクエストのログ記録 を参照してください。

また、s3-bucket-logging-enabled AWS Config マネージドルールを使用する継続的な検出コントロールの実装を検討してください。

AWS CloudTrail の使用

AWS CloudTrail は、Amazon S3 のユーザー、ロール、または AWS のサービス によって実行されたアクションのレコードを提供します。CloudTrail によって収集されたデータを使用して、以下の情報を判断できます。

- Amazon S3 に対して行われたリクエスト
- リクエストが行われた IP アドレス
- リクエストを行ったユーザー
- リクエストが行われた時間
- リクエストに関するその他の詳細

例えば、データアクセスに影響する PUT アクション (特に `PutBucketAcl`、`PutObjectAcl`、`PutBucketPolicy` および `PutBucketWebsite`) の CloudTrail エントリを特定できます。

AWS アカウントをセットアップすると、CloudTrail はデフォルトで有効になっています。CloudTrail コンソールで最近のイベントを確認できます。Amazon S3 バケットのアクティビティとイベントの継続的なレコードを作成するには、CloudTrail コンソールで証跡を作成できます。詳細については、「AWS CloudTrail ユーザーガイド」の「データイベントをログ記録する」を参照してください。

証跡を作成する際に、データイベントをログ記録するように CloudTrail を設定できます。データイベントは、リソース上またはリソース内で実行されたリソースオペレーションのレコードです。Amazon S3 では、データイベントは、個々のバケットのオブジェクトレベルの API アクティビティを記録します。CloudTrail は、`GetObject`、`DeleteObject`、`PutObject` などの Amazon S3 オブジェクトレベルの API オペレーションのサブセットをサポートしています。CloudTrail と Amazon S3 との連携の詳細については、AWS CloudTrail を使用した Amazon S3 API コールのログ記録 を参照してください。Amazon S3 コンソールでは、S3 バケットとオブジェクトの CloudTrail イベントログ記録の有効化 に S3 バケットを設定することもできます。

AWS Config に用意されている管理ルール (`cloudtrail-s3-dataevents-enabled`) を使用すると、少なくとも 1 つの CloudTrail 証跡が S3 バケットのデータイベントのログに記録していることを確認できます。詳細については、cloudtrail-s3-dataevents-enabled デベロッパーガイドのAWS Config を参照してください。

AWS Config の有効化

このトピックに示すベストプラクティスのいくつかでは、AWS Config ルールの作成を提案しています。AWS Config では、AWS リソースの設定を評価、監査、診断するのに役立ちます。AWS Config では、リソースの設定をモニタリングし、目的とする安全な設定に対して、記録された設定を評価できます。AWS Config では、次のことを実行できます。

- 設定の変更と AWS リソース間の関係を確認します。
- 詳細なリソース設定履歴の調査
- 内部ガイドラインに指定されている設定に対して全体的なコンプライアンスを決定します。

AWS Config を使用すると、コンプライアンス監査、セキュリティ分析、変更管理、運用上のトラブルシューティングを簡素化できます。詳細については、AWS Config デベロッパーガイドのコンソールを使用した AWS Config の設定 を参照してください。記録するリソースタイプを指定するときは、必ず Amazon S3 リソースを含めてください。

> **Important**
>
> AWS Config マネージドルールは Amazon S3 リソースを評価する際に汎用バケットのみをサポートします。AWS Config ディレクトリバケットの設定変更は記録されません。

詳細については、「AWS Config デベロッパーガイド」の「AWS Config マネージドルール」と「AWS Config マネージドルールリスト」を参照してください。

AWS Config を使用する方法の例については、「AWS セキュリティブログ」の「パブリックアクセスを許可する Amazon S3 バケットを AWS Config でモニタリングおよび応答する方法」を参照してください。

Amazon Macie を使用した機密データの検出

Amazon Macie は、機械学習とパターンマッチングを使用して機密データを発見するセキュリティサービスです。Macie はデータセキュリティリスクを可視化し、それらのリスクに対する自動保護を可能にします。Macie を使用すると、Amazon S3 データアセット内の機密データの検出とレポートを自動化して、組織が Amazon S3 に保存しているデータを詳細に把握できます。

Macie を使用して機密データを検出するには、多くの国または地域の機密データタイプの大規模かつ増加しているリストを検出するように設計された組み込み型の基準と手法を使用できます。これらの機密データタイプには、複数の種類の個人を特定できる情報 (PII)、財務データ、認証情報データが含まれます。一致するテキストパターンを定義する正規表現を使用したり、オプションで結果を絞り込む文字シーケンスや近接ルールなど、自分で定義したカスタム基準を使用することもできます。

Macie が S3 オブジェクト内の機密データを検出すると、Macie はセキュリティ上の検出結果を生成して通知します。この検出結果により、影響を受けたオブジェクト、Macie が見つけた機密データの種類と出現回数、および影響を受けた S3 バケットとオブジェクトの調査に役立つ追加情報が得られます。詳細については、「Amazon Macie ユーザーガイド」を参照してください。

S3 ストレージレンズの使用

S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージ分析機能です。また、S3 Storage Lens は、メトリクスを分析して、ストレージコストを最適化し、データ保護に関するベストプラクティスを適用するために使用できるコンテキストに応じた推奨事項を提供します。

S3 Storage Lens で、組織全体でどれだけのストレージがあるか、または最も急速に成長しているバケットとプレフィックスは何かなどの、要約されたインサイトを生成できます。S3 Storage Lens メトリクスを使用して、コスト最適化の機会を特定し、データ保護とアクセス管理のベストプラクティスを実装し、アプリケーションワークロードのパフォーマンスを向上させることができます。

例えば、S3 ライフサイクルルールがないバケットを特定して、7 日以上経過した不完全なマルチパートアップロードを中止できます。また、S3 レプリケーションや S3 バージョニングの使用など、データ保護のベストプラクティスに従っていないバケットを特定することもできます。詳細については、「Amazon S3 ストレージレンズについて」を参照してください。

AWS セキュリティアドバイザリをモニタリングする

AWS アカウント について Trusted Advisor に投稿されたセキュリティ勧告を定期的に確認することをお勧めします。特に、「オープンアクセス許可」のある Amazon S3 バケットに関する警告を探します。このステップは、describe-trusted-advisor-checks を使用して、プログラムで実行できます。

さらに、各 AWS アカウント に登録されているメインの E メールアドレスを注意してモニタリングしてください。AWS は、この E メールアドレスを使用して、ユーザーに影響を与える可能性のあるセキュリティ問題が発生した場合に連絡します。

広範な影響を与える AWS の運用上の問題は AWS Health Dashboard - Service health に投稿されます。運用上の問題も、AWS Health Dashboard を通じて個々のアカウントに投稿されます。詳細については、「AWS Healthドキュメント」を参照してください。

マネージド AWS セキュリティサービスによるデータセキュリティのモニタリング

いくつかのマネージド AWS セキュリティサービスは、Amazon S3 データのセキュリティとコンプライアンスのリスクを特定、評価、監視するのに役立ちます。また、こうしたリスクからデータを保護するのにも役立ちます。これらのサービスには、単一の AWS アカウント から数千の AWS アカウント にまたがる組織向けのリソースに拡張できるように設計された、自動検知、モニタリング、保護機能が含まれます。

AWS 検出および対応サービスを使うと、潜在的なセキュリティ設定ミス、脅威、または予期しない動作を特定できるため、環境内の不正または悪意のある可能性があるアクティビティに対して迅速に対応できます。AWS データ保護サービスは、データ、アカウント、ワークロードをモニタリングし、不正アクセスから保護するのに役立ちます。また、Amazon S3 のデータ資産内で個人を特定できる情報 (PII) などの機密データを発見するのにも役立ちます。

データセキュリティとコンプライアンスのリスクを特定して評価するのをサポートするため、マネージド AWS セキュリティサービスでは、検出結果を生成して、Amazon S3 データに関する潜在的なセキュリティイベントや問題を通知します。検出結果には関連する詳細は、インシデント対応のワークフローとポリシーに従ってこれらのリスクを調査、評価、対処するために使用できます。各サービスを使用して、検出結果データに直接アクセスできます。また、セキュリティインシデントやイベント管理システム (SIEM) など、他のアプリケーション、サービス、システムにデータを送信することもできます。

Amazon S3 データのセキュリティをモニタリングするには、これらのマネージド AWS セキュリティサービスの使用を検討してください。

Amazon GuardDuty

Amazon GuardDuty は、悪意のあるアクティビティがないか継続的に AWS アカウント をモニタリングし、詳細なセキュリティ検出結果を提供する脅威検知サービスです。

GuardDuty の S3 保護機能を使用すると、Amazon S3 リソースの AWS CloudTrail 管理イベントとデータイベントを分析するように GuardDuty を設定できます。それにより、GuardDuty は、悪意のあるアクティビティや疑わしいアクティビティについてこれらのイベントをモニタリングします。分析を行い、潜在的なセキュリティリスクを特定するため、GuardDuty は脅威インテリジェンスフィードと機械学習を使用しています。

GuardDuty では、Amazon S3 リソースについてさまざまな種類のアクティビティをモニタリングできます。例えば、Amazon S3 の CloudTrail 管理イベント

には、ListBuckets、DeleteBucket、PutBucketReplication など、バケットレベルのオペレーションが含まれます。Amazon S3 のデータイベントには、GetObject、ListObjects、PutObject などのオブジェクトレベルのオペレーションが含まれます。GuardDuty が異常なアクティビティや潜在的に悪意のあるアクティビティを検出すると、検出結果を生成してユーザーに通知します。

詳細については、「Amazon GuardDuty ユーザーガイド」の「Amazon GuardDuty での Amazon S3 の保護」を参照してください。

Amazon Detective

Amazon Detective は調査プロセスを簡素化し、セキュリティ調査をより迅速かつ効果的に実施できるようにします。Detective が提供する事前に作成されたデータ集計、要約、およびコンテキストは、考えられるセキュリティ問題の性質と範囲を迅速に分析および評価するのに役立ちます。

Detective は、AWS CloudTrail からの API 呼び出しや AWS の Amazon VPC フローログなど、時間ベースのイベントを自動的に抽出します。また、Amazon GuardDuty によって生成された検出結果も取り込みます。次に Detective は、機械学習、統計分析、グラフ理論を使用して、セキュリティ調査をより迅速かつ効率的に行うのに役立つビジュアライゼーションを生成します。

これらのビジュアライゼーションは、リソースの動作と時間の経過に伴うそれらのインタラクションに関するインタラクティブな統合ビューを提供します。この動作グラフを詳しく確認すると、失敗したログオン試行や疑わしい API コールなど、さまざまなアクションを調べることができます。また、これらのアクションが S3 バケットやオブジェクトなどのリソースにどのように影響するかを確認できます。

詳細については、「Amazon Detective 管理ガイド」を参照してください。

IAM Access Analyzer

AWS Identity and Access Management Access Analyzer (IAM Access Analyzer) は、外部エンティティと共有されているリソースを識別するのに役立ちます。また、IAM Access Analyzer を使用して IAM ポリシーをポリシー文法やベストプラクティスに照らして検証し、AWS CloudTrail ログのアクセスアクティビティに基づいて IAM ポリシーを生成することもできます。

IAM Access Analyzer は、ロジックベースの推論を使用して、バケットポリシーなど、AWS 環境内のリソースポリシーを分析します。IAM Access Analyzer for S3 は、インターネットの任意のユーザーや他の AWS アカウント (組織外のアカウントを含む) にアクセスを許可するように S3 バケットが設定された場合、警告します。例えば、IAM Access Analyzer for S3 は、バケットのアクセスコントロールリスト (ACL)、バケットポリシー、マルチリージョンアクセスポイントポリシー、またはアクセスポイントポリシーを通じて、バケットに読み取りまたは書き込みのアク

セス権が提供されていることを報告する場合があります。パブリックバケットまたは共有バケットごとに、パブリックアクセスや共有アクセスのソースとレベルを示す検出結果が送信されます。この情報を用いて、迅速で正確な是正処置を講じ、バケットへのアクセスを意図したとおりに復元できます。

詳細については、「IAM Access Analyzer for S3 を使用したバケットアクセスの確認」を参照してください。

Amazon Macie

Amazon Macie は、機械学習とパターンマッチングを使用して機密データを検出し、データセキュリティリスクを可視化し、それらのリスクに対する自動保護を可能にするデータセキュリティサービスです。

Macie を使用すると、S3 データバケット内の機密データの検出とレポートを自動化でき、組織が Amazon S3 に保存しているデータを詳細に把握できるようになります。機密データを検出するには、Macie が提供する組み込み型の基準と手法、ユーザーが定義するカスタム基準、またはこの 2 つの組み合わせを使用できます。Macie が S3 オブジェクト内の機密データを検出すると、Macie は検出結果を生成して通知します。この検出結果により、影響を受けたバケットとオブジェクト、Macie が見つけた機密データの種類と出現回数、および調査を円滑に進めるのに役立つ追加の詳細に関する情報が得られます。

Macie は、Amazon S3 データのセキュリティ体制の分析情報を提供する統計やその他のデータも提供し、セキュリティおよびアクセスコントロールのために S3 バケットを自動的に評価およびモニタリングします。Macie は、バケットがパブリックアクセス可能になっているなど、データのセキュリティまたはプライバシーに関する潜在的な問題を検出した場合、必要に応じて確認および修正するための検出結果を生成します。

詳細については、「Amazon Macie ユーザーガイド」を参照してください。

AWS Security Hub

AWS Security Hub は、セキュリティのベストプラクティスをチェックし、複数のソースからのアラートと検出結果を 1 つの形式に集約し、自動修復を可能にするセキュリティ体制管理サービスです。

Security Hub は、Amazon Detective、Amazon GuardDuty、IAM Access Analyzer、Amazon Macie などの統合 AWS Partner Network セキュリティソリューションおよび AWS のサービスからセキュリティ検出結果データを収集して提供します。また、AWS ベストプラクティスとサポートされている業界標準に基づいて、継続的な自動セキュリティチェックを実行することによって、独自の検出結果を生成します。

Security Hub はその後、プロバイダー間で結果を関連づけて統合し、最も重要な検出結果を優先化し、処理できるようにします。また、カスタムアクションもサポートされています。カスタムアクションを使用して、特定クラスの検出結果に対する応答または修復アクションを呼び出すことができます。

Security Hub を使用すると、Amazon S3 リソースのセキュリティとコンプライアンスのステータスを評価できます。これは、個々の AWS リージョン と複数のリージョンで、組織のセキュリティ体制に対するより広範な分析の一部として行うことができます。これには、セキュリティの傾向分析や最も優先度の高いセキュリティ問題の特定が含まれます。また、複数の AWS リージョン からの検出結果を集約したり、1 つのリージョンから集計した検出結果データをモニタリングおよび処理することもできます。

詳細については、「AWS Security Hub ユーザーガイド」の「Amazon Simple Storage Service コントロール」を参照してください。

Amazon S3 ストレージの管理

Amazon S3 でバケットを作成してオブジェクトをアップロードすると、バージョニング、ストレージクラス、オブジェクトロック、バッチ操作、レプリケーション、タグその他のさまざまな機能を使って、オブジェクトストレージを管理できるようになります。以下のセクションでは、Amazon S3 で利用できるストレージの管理機能や特徴について、詳しく説明します。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

トピック

S3 バケットでのバージョニングの使用

Amazon S3 のバージョニングとは、同じバケット内でオブジェクトの複数のバリアントを保持する手段のことです。S3 のバージョニング機能を使用すると、バケットに保存されたすべてのオブジェクトのすべてのバージョンを、保存、取得、復元することができます。バージョニングを使用すれば、意図しないユーザーアクションからもアプリケーション障害からも、より簡単に復旧できます。バケットのバージョニングを有効にすると、Amazon S3 が同じオブジェクトに対する複数の書き込みリクエストを同時に受信した場合に、すべてのオブジェクトが保存されます。

バージョニングを有効にしたバケットは、オブジェクトを誤って削除したり上書きしたりしても、復元が簡単に行えます。例えば、オブジェクトを削除した場合、Amazon S3 は、オブジェクトを完全に削除する代わりに削除マーカーを挿入します。その削除マーカーが、最新のオブジェクトバージョンになります。オブジェクトを上書きすると、バケット内の新しいオブジェクトバージョンになります。いつでも以前のバージョンを復元できます。詳細については、「バージョニングが有効なバケットからのオブジェクトバージョンの削除」を参照してください。

デフォルトではバケットの S3 バージョニングは無効になっているので、明示的に有効にする必要があります。詳細については、「バケットでのバージョニングの有効化」を参照してください。

> **Note**
>
> - SOAP API は、S3 バージョニングをサポートしていません。SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。
> - 通常の Amazon S3 料金は、保存または移行されるオブジェクトのバージョンごとに適用されます。オブジェクトの各バージョンはオブジェクト全体であり、以前のバージョンの単なる差分ではありません。したがって、オブジェクトの 3 つのバージョンを保存している場合、3 つのオブジェクトに対して課金されます。

バージョニングが無効なバケット、有効なバケット、停止されているバケット

バケットは、次の 3 つの状態のいずれかになります。

- Unversioned (バージョニングが無効) (デフォルト)
- Versioning-enabled (バージョニングが有効)

- Versioning-suspended (バージョニングが停止)

バージョニングは、バケットレベルで有効化および停止します。一度バケットのバージョニングを有効にすると、バージョニング無効の状態に戻すことはできません。ただし、そのバケットのバージョニングを一時停止することは可能です。

バージョニング状態は、バケット内の一部ではなくすべてのオブジェクトに適用されます。バケットでバージョニングを有効にすると、すべての新しいオブジェクトがバージョニングされ、一意のバージョン ID が割り当てられます。バージョニングが有効化された時点でバケット内に既に存在していたオブジェクトは、それ以降常にバージョニングされ、将来のリクエストによって変更されたときに一意のバージョン ID が割り当てられます。次の点に注意してください。

- バージョニング状態を設定する前にバケットに保存したオブジェクトのバージョン ID は `null` です。バージョニングを有効にした場合、バケットに含まれる既存のオブジェクトは変更されません。変更されるのは、Amazon S3 が今後のリクエストでオブジェクトを処理する方法です。詳細については、「バージョニングが有効なバケットでのオブジェクトの操作」を参照してください。
- バケット所有者 (または適切なアクセス許可を持つユーザー) は、バージョニングを停止してオブジェクトバージョンの生成を中断できます。バージョニングを停止しても、バケットに含まれる既存のオブジェクトは変更されません。変更されるのは、Amazon S3 が今後のリクエストでオブジェクトを処理する方法です。詳細については、「バージョニングが停止されたバケットのオブジェクトの操作」を参照してください。

S3 ライフサイクルでの S3 バージョニングの使用

データ保持のアプローチやストレージコストの管理をカスタマイズするには、S3 ライフサイクルでオブジェクトバージョニングを使用します。詳細については、「ストレージのライフサイクルの管理」を参照してください。AWS Management Console、AWS CLI、AWS SDK、または REST API を使って S3 ライフサイクル設定を作成する方法の詳細は、「バケットにライフサイクル設定を設定する」を参照してください。

> **Important**
>
> バージョニングが無効であるバケットに、オブジェクトの有効期限ライフサイクル設定があり、有効化したときに同一の完全な削除動作を維持させたいときは、以前の有効期限切れ設定を追加する必要があります。以前の有効期限切れのライフサイクル設定は、バージョニングが有効なバケットで、以前のオブジェクトバージョンの削除を管理します (バージョニングが有効なバケットは、1 個の最新のオブジェクトバージョン と、0 個以上の以前のバー

ジョンを維持します)。詳細については、「バケットにライフサイクル設定を設定する」を参照してください。

S3 バージョニングの操作に関する詳細は、以下のトピックを参照してください。

トピック

- S3 バージョニングの仕組み
- バケットでのバージョニングの有効化
- MFA 削除の設定
- バージョニングが有効なバケットでのオブジェクトの操作
- バージョニングが停止されたバケットのオブジェクトの操作

S3 バージョニングの仕組み

S3 バージョニングを使用すると、オブジェクトの複数のバージョンを 1 つのバケットに保持し、誤って削除または上書きされたオブジェクトを復元できます。例えば、S3 バージョニングをバケットに適用すると、次の変更が行われます。

- オブジェクトを削除した場合、Amazon S3 では、オブジェクトを完全に削除する代わりに削除マーカーを挿入し、それが最新のオブジェクトバージョンになります。これにより、以前のバージョンを復元できます。詳細については、「バージョニングが有効なバケットからのオブジェクトバージョンの削除」を参照してください。
- オブジェクトを上書きすると、Amazon S3 のバケットに新しいオブジェクトバージョンが追加されます。以前のバージョンはバケットに残りますが、最新のバージョンではなくなります。以前のバージョンは復元することができます。

> **Note**
>
> 通常の Amazon S3 料金は、保存または移行されるオブジェクトのバージョンごとに適用されます。オブジェクトの各バージョンはオブジェクト全体であり、以前のバージョンの差分ではありません。したがって、オブジェクトの 3 つのバージョンを保存している場合、3 つのオブジェクトに対して課金されます。

作成する各 S3 バケットには、それに関連付けられたバージョニングのサブリソースがあります。(詳しくは、バケット設定オプション を参照してください)。デフォルトでは、バケットのバージョニングは無効で、バージョニングのサブリソースには、以下のとおり、空のバージョニング設定が保存されます。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</VersioningConfiguration>
```

バージョニングを有効にするには、Enabled ステータスを含むバージョニング設定を使用して、Amazon S3 にリクエストを送信します。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

バージョニングを停止するには、ステータス値を Suspended に設定します。

> **Note**
>
> バケットで初めてバージョニングを有効にした場合、変更が完全に反映されるまでに少し時間がかかることがあります。バケットのオブジェクトへの書き込みオペレーション (PUT または DELETE) は、バージョニングを有効にして 15 分待ってから発行することをお勧めします。

バケット所有者とすべての承認済み AWS Identity and Access Management (IAM) ユーザーは、バージョニングを有効にすることができます。バケット所有者はバケットを作成する AWS アカウントです。権限の詳細については、Amazon S3 用 Identity and Access Management をご参照ください。

AWS Management Console、AWS Command Line Interface (AWS CLI)、または REST API を使用した S3 バージョニングの有効化と無効化の詳細については、「the section called “バケットでのバージョニングの有効化”」を参照してください。

トピック

- バージョン ID
- バージョニングのワークフロー

バージョン ID

バケットのバージョニングを有効にすると、Amazon S3 は、保存されるオブジェクトに対して一意のバージョン ID を自動的に生成します。例えば、1 つのバケット内に、photo.gif (バージョン 111111) と photo.gif (バージョン 121212) のように、同じキー (オブジェクト名) でもバージョン ID が異なる 2 つのオブジェクトを保持することができます。
キーは同じでもバージョン ID が異なる 2 つのオブジェクトがある、バージョニングが有効になっているバケットを示す図。

S3 バージョニングが有効になっているかどうかにかかわらず、各オブジェクトにはバージョン ID があります。S3 バージョニングが有効にされていない場合、Amazon S3 はバージョン ID の値を null に設定します。S3 バージョニングを有効にした場合、Amazon S3 がオブジェクトにバージョン ID 値を割り当てます。この値により、そのオブジェクトが同じキーの他のバージョンと区別されます。

既存のバケットで S3 バージョニングを有効にしても、バケットにすでに保存されているオブジェクトは変更されません。バージョン ID (null)、コンテンツ、アクセス権限が変更されることはありません。S3 バージョニングを有効にすると、バケットに追加された各オブジェクトにバージョン ID が与えられ、同じキーの他のバージョンと区別されます。

バージョン ID を生成できるのは Amazon S3 のみです。また、編集はできません。バージョン ID は、Unicode、UTF-8 エンコード、URL 対応の不透明な文字列で、長さは 1,024 バイト以下です。次に例を示します。

```
3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo
```

> **Note**
>
> 簡素化のため、本トピックの他の例ではさらに短い ID を使用します。

バージョニングのワークフロー

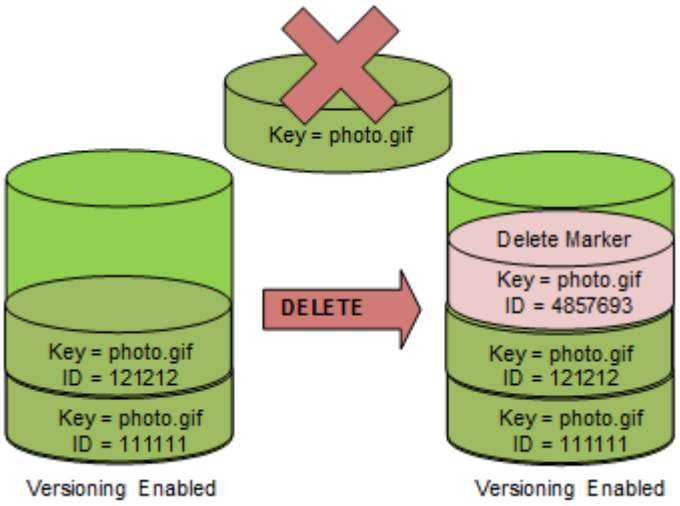
バージョニングが有効になっているバケットにオブジェクトを PUT しても、以前のバージョンは上書きされません。次の図に示すように、既に同じ名前のオブジェクトが入っているバケットに新しいバージョンの photo.gif が PUT された場合、以下の挙動が生じます。

- 元のオブジェクト (ID = 111111) はバケットに残ります。

- Amazon S3 は新しいバージョン ID (121212) を生成し、この新しいバージョンのオブジェクトをバケットに追加します。

この機能を使用すると、オブジェクトが誤って上書きまたは削除された場合に、オブジェクトの以前のバージョンを取得できます。

オブジェクトの DELETE を実行すると、次の図に示すように、バケット内のすべてのバージョンが残り、Amazon S3 によって削除マーカーが挿入されます。



削除マーカーはオブジェクトの最新バージョンになります。デフォルトで、GET リクエストは最後に保存されたバージョンを取得します。最新バージョンが削除マーカーである場合に GET Object リクエストを実行すると、次の図に示すように 404 Not Found エラーが返されます。

ただし、バージョン ID を指定すれば、オブジェクトの以前のバージョンを GET することができます。次の図では、特定のオブジェクトバージョン 111111 の GET を実行します。これはオブジェクトの最新バージョンではありませんが、Amazon S3 はこのバージョンを返します。

詳細については、「バージョニングが有効なバケットからのオブジェクトバージョンの取得」を参照してください。

オブジェクトを完全に削除するには、削除するバージョンを指定します。Amazon S3 バケット所有者または権限のある IAM ユーザーのみが、特定のバージョンを永久に削除することができます。DELETE オペレーションで versionId を指定すると、そのオブジェクトバージョンは完全に削除され、Amazon S3 による削除マーカーの挿入も行われません。

多要素認証 (MFA) Delete に対応するようにバケットを設定すると、セキュリティを強化できます。多要素認証 (MFA) Delete を有効にすると、バケット所有者は特定のバージョンを削除したりバケッ

トのバージョニング状態を変更したりするために、すべてのリクエストに 2 つの認証形態を含める必要があります。詳細については、「MFA 削除の設定」を参照してください。

オブジェクトの新しいバージョンが作成されるのはどのような場合か

オブジェクトの新しいバージョンは、新しいオブジェクトを PUT した場合にのみ作成されます。CopyObject のような特定のアクションは、PUT オペレーションを実装することにより機能することに注意してください。

現在のオブジェクトを変更するアクションを実行しても、新しいオブジェクトを PUT しないため、新しいバージョンは作成されません。これには、オブジェクトのタグの変更などのアクションが含まれます。

> **Important**
>
> S3 バージョニングを有効にしたバケットへの Amazon S3 PUT または DELETE オブジェクトリクエストに対して受信される HTTP 503 (Service Unavailable) レスポンスの数が著しく増加した場合、バケットに数百万のバージョンが存在するオブジェクトがある可能性があります。詳細については、トラブルシューティングの「S3 バージョニングの使用」セクションを参照してください。

バケットでのバージョニングの有効化

S3 バージョニングを使用すると、1 つのバケットで複数バージョンのオブジェクトを維持できます。このセクションでは、コンソール、REST API、AWS SDK、AWS Command Line Interface (AWS CLI) を使って、バケットでバージョニングを有効にする方法の例を説明します。

> **Note**
>
> バケットで初めてバージョニングを有効にしたときは、変更が完全に反映されるまでに、最長で 15 分かかることがあります。バケットへのオブジェクトの書き込みオペレーション (PUT または DELETE) は、バージョニングを有効にして 15 分待ってから発行することをお勧めします。この変換が完了する前に発行された書き込みオペレーションが、バージョン管理されていないオブジェクトに適用される場合があります。

S3 バージョニングの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。バージョニングが有効になっているバケットでの、オブジェクトの操作に関する詳細は、「バージョニングが有効なバケットでのオブジェクトの操作」を参照してください。

S3 バージョニングを使用してデータを保護する方法の詳細については、「Tutorial: Protecting data on Amazon S3 against accidental deletion or application bugs using S3 Versioning, S3 Object Lock, and S3 Replication」(チュートリアル: S3 バージョニング、S3 オブジェクトロック、S3 レプリケーションを使用して、Amazon S3 上のデータを予期しない削除やアプリケーションのバグから保護する) を参照してください。

作成する各 S3 バケットには、それに関連付けられたバージョニングのサブリソースがあります。(詳しくは、バケット設定オプション を参照してください)。デフォルトでは、バケットのバージョニングは無効で、バージョニングのサブリソースには、以下のとおり、空のバージョニング設定が保存されます。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</VersioningConfiguration>
```

バージョニングを有効にするには、状態を含むバージョニング設定を使用して、Amazon S3 にリクエストを送信します。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

バージョニングを停止するには、ステータス値を Suspended に設定します。

バケット所有者とすべての承認されたユーザーは、バージョニングを有効にすることができます。バケット所有者は、バケットを作成した AWS アカウント (ルートアカウント) です。権限の詳細については、Amazon S3 用 Identity and Access Management をご参照ください。

以下のセクションでは、コンソール、AWS CLI、AWS SDK を使って S3 バージョニングを有効にする方法の詳細を説明します。

S3 コンソールの使用

AWS Management Console を使用して、S3 バケットでバージョニングを有効にするには、次の手順に従います。

S3 バケットのバージョニングを有効または無効にするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] のリストで、バージョニングを有効にするバケットの名前を選択します。
3. [プロパティ] を選択します。
4. [バケットのバージョニング] で [編集] を選択します。
5. [中断] または [有効化] を選択し、[変更を保存] を選択します。

> **Note**
>
> バージョニングで AWS 多要素認証 (MFA) を使用できます。バージョニングに MFA を使用しているときに、オブジェクトバージョンを完全に削除したり、バージョニングを停止または再有効化したりする場合は、AWS アカウント のアクセスキーと有効なコードを、アカウントの MFA デバイスから指定することが必要になります。
>
> バージョニングで MFA を使用するには、`MFA Delete` を有効にします。ただし、AWS Management Console を使用して `MFA Delete` を有効にすることはできません。AWS Command Line Interface (AWS CLI) または API を使用する必要があります。詳細については、「MFA 削除の設定」を参照してください。

AWS CLI の使用

次の例では、S3 バケットでバージョニングを有効にします。

```
aws s3api put-bucket-versioning --bucket example-s3-bucket1 --versioning-configuration
 Status=Enabled
```

次の例では、バケットで S3 バージョニングと多要素認証 (MFA) 削除を有効にします。

```
aws s3api put-bucket-versioning --bucket example-s3-bucket1 --versioning-configuration
 Status=Enabled,MFADelete=Enabled --mfa "SERIAL 123456"
```

> **Note**
>
> MFA 削除を使用するときは、承認済みの物理または仮想の認証デバイスが必要です。Amazon S3 での MFA 削除の使用に関する詳細は、「MFA 削除の設定」を参照してください。

AWS CLI を使用したバージョニングの有効化に関する詳細は、AWS CLI CLI コマンドリファレンスの「put-bucket-versioning」を参照してください。

AWS SDK の使用

次の例では、バケットでバージョニングを有効にし、AWS SDK for Java と AWS SDK for .NET を使ってバージョニングの状態を復元します。他の AWS SDK の使用の詳細については、「AWS デベロッパーセンター」を参照してください。

.NET

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using System;
using Amazon.S3;
using Amazon.S3.Model;

namespace s3.amazon.com.docsamples
{
    class BucketVersioningConfiguration
    {
        static string bucketName = "*** bucket name ***";

        public static void Main(string[] args)
        {
            using (var client = new AmazonS3Client(Amazon.RegionEndpoint.USEast1))
            {
                try
                {
                    EnableVersioningOnBucket(client);
                    string bucketVersioningStatus =
 RetrieveBucketVersioningConfiguration(client);
                }
                catch (AmazonS3Exception amazonS3Exception)
```

```
                {
                    if (amazonS3Exception.ErrorCode != null &&
                        (amazonS3Exception.ErrorCode.Equals("InvalidAccessKeyId")
                        ||
                        amazonS3Exception.ErrorCode.Equals("InvalidSecurity")))
                    {
                        Console.WriteLine("Check the provided AWS Credentials.");
                        Console.WriteLine(
                        "To sign up for service, go to http://aws.amazon.com/s3");
                    }
                    else
                    {
                        Console.WriteLine(
                         "Error occurred. Message:'{0}' when listing objects",
                         amazonS3Exception.Message);
                    }
                }
            }

            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }

        static void EnableVersioningOnBucket(IAmazonS3 client)
        {

                PutBucketVersioningRequest request = new PutBucketVersioningRequest
                {
                    BucketName = bucketName,
                    VersioningConfig = new S3BucketVersioningConfig
                    {
                        Status = VersionStatus.Enabled
                    }
                };

                PutBucketVersioningResponse response =
client.PutBucketVersioning(request);
        }


        static string RetrieveBucketVersioningConfiguration(IAmazonS3 client)
        {
                GetBucketVersioningRequest request = new GetBucketVersioningRequest
                {
```

```
                    BucketName = bucketName
                };

                GetBucketVersioningResponse response =
 client.GetBucketVersioning(request);
                return response.VersioningConfig.Status;
            }
    }
}
```

Java

作業サンプルの作成方法およびテスト方法については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.SetBucketVersioningConfigurationRequest;

public class BucketVersioningConfigurationExample {
    public static String bucketName = "*** bucket name ***";
    public static AmazonS3Client s3Client;

    public static void main(String[] args) throws IOException {
        s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
        s3Client.setRegion(Region.getRegion(Regions.US_EAST_1));
        try {

            // 1. Enable versioning on the bucket.
         BucketVersioningConfiguration configuration =
           new BucketVersioningConfiguration().withStatus("Enabled");

   SetBucketVersioningConfigurationRequest setBucketVersioningConfigurationRequest
 =
     new SetBucketVersioningConfigurationRequest(bucketName,configuration);
```

```
 s3Client.setBucketVersioningConfiguration(setBucketVersioningConfigurationRequest);

   // 2. Get bucket versioning configuration information.
   BucketVersioningConfiguration conf =
 s3Client.getBucketVersioningConfiguration(bucketName);
    System.out.println("bucket versioning configuration status:    " +
 conf.getStatus());

        } catch (AmazonS3Exception amazonS3Exception) {
            System.out.format("An Amazon S3 error occurred. Exception: %s",
 amazonS3Exception.toString());
        } catch (Exception ex) {
            System.out.format("Exception: %s", ex.toString());
        }
    }
}
```

Python

次の Python コードの例では、Amazon S3 バケットを作成し、バージョニング用に有効にして、オブジェクトの最新ではないバージョンが 7 日後に失効するライフサイクルを設定しています。

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
```

```
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
```

```
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

MFA 削除の設定

Amazon S3 バケットで S3 バージョニングを行うときに、MFA (多要素認証) Delete が有効になるようにバケットを設定すれば、セキュリティをさらに強化できます。この設定を行うと、バケット所有者は、特定のバージョンを削除したりバケットのバージョニング状態を変更したりするリクエストに、2 つの認証形式を含めることが必要になります。

MFA Delete では、以下のいずれかの操作で追加の認証が必要になります。

- バケットのバージョニング状態を変更する
- オブジェクトバージョンを完全に削除する

MFA Delete では、2 つの認証形式の組み合わせが必要になります。

- セキュリティ認証情報
- 有効なシリアル番号、スペース、および承認済みの認証デバイスに表示される 6 桁のコードを連結した文字

MFA Delete は、このようにして、認証情報に不正なアクセスがあった場合などにセキュリティを強化します。MFA 削除は、削除アクションを開始したユーザーに MFA コードを使って MFA デバイスの物理的所有を証明するように要求したり、削除アクションに摩擦とセキュリティのレイヤーをさらに追加したりすることで、バケットの偶発的な削除を防ぎます。

MFA 削除が有効になっているバケットを特定するには、Amazon S3 ストレージレンズメトリクスを使用できます。S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージの分析機能です。詳細については、「S3 Storage Lens を使用したストレージのアクティビティと使用状況の評価」を参照してください。メトリクスの完全なリストについては、「S3 ストレージレンズメトリクスに関する用語集」を参照してください。

バケット所有者、バケットを作成した AWS アカウント (ルートアカウント)、およびすべての承認されたユーザーは、バージョニングを有効にすることができます。ただし、MFA Delete を有効化できるのは、バケット所有者 (ルートアカウント) のみです。詳細については、AWS セキュリティブログの「MFA を使用した AWS へのアクセスの保護」を参照してください。

> **Note**
>
> バージョニングで MFA Delete を使用するには、`MFA Delete` を有効にします。ただし、AWS Management Console を使用して `MFA Delete` を有効にすることはできません。AWS Command Line Interface (AWS CLI) または API を使用する必要があります。
> バージョニングで MFA Delete を使用する例については、トピック バケットでのバージョニングの有効化 の具体例のセクションを参照してください。
> ライフサイクル設定で MFA 削除を使用することはできません。ライフサイクル設定と、それらが他の設定とやり取りする方法の詳細については、「ライフサイクルとその他のバケット設定」を参照してください。

MFA 削除を有効または無効にするには、バケットのバージョニングの設定に使用するものと同じ API を使用します。Amazon S3 では、バケットのバージョニング状態を格納しているものを同じバージョニングサブリソースに、MFA Delete の設定が格納されます。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>VersioningState</Status>
  <MfaDelete>MfaDeleteState</MfaDelete>
</VersioningConfiguration>
```

MFA Delete を使用するときは、ハードウェアデバイスまたは仮想 MFA デバイスを使用して認証コードを生成します。次の例は、生成された認証コードがハードウェアデバイスに表示されている様子を示しています。

MFA Delete と MFA で保護された API アクセスは、異なるシナリオで保護を行うことを目的とした機能です。バケットに MFA Delete を設定することで、バケット内のデータが誤って削除されることのないようにします。MFA で保護された API アクセスは、Amazon S3 の機密リソースにアクセスする場合に、別の認証要素 (MFA コード) を適用するために使用します。これらの Amazon S3 リ

ソースに対するすべてのオペレーションが、MFA を使用した一時証明書によって実行されるように要求できます。例については、「MFA が必要」を参照してください。

認証デバイスの購入およびアクティベートの方法の詳細については、「多要素認証」を参照してください。

S3 バージョニングを有効にして MFA 削除を設定するには

AWS CLI の使用

次の例では、バケットで S3 バージョニングと多要素認証 (MFA) 削除を有効にします。

```
aws s3api put-bucket-versioning --bucket example-s3-bucket1 --versioning-configuration
 Status=Enabled,MFADelete=Enabled --mfa "SERIAL 123456"
```

REST API の使用

Amazon S3 REST API を使用したルーティングルールの設定の詳細については、Amazon Simple Storage Service API リファレンスの「PutBucketVersioning」を参照してください。

バージョニングが有効なバケットでのオブジェクトの操作

バージョニング状態を設定する前に Amazon S3 バケットに保存されたオブジェクトのバージョン ID は、null です。バージョニングを有効にした場合、バケットに含まれる既存のオブジェクトは変更されません。変更されるのは、Amazon S3 が今後のリクエストでオブジェクトを処理する方法です。

オブジェクトバージョンの移行

明確なライフサイクルが定義されているオブジェクトに対してライフサイクル設定ルールを定義することにより、オブジェクトの有効期間内の特定の時点でオブジェクトバージョンを S3 Glacier Flexible Retrieval ストレージクラスに移行できます。詳細については、「ストレージのライフサイクルの管理」を参照してください。

このセクションのトピックでは、バージョニングが有効なバケットでのさまざまなオブジェクトのオペレーションについて説明します。バージョニングの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

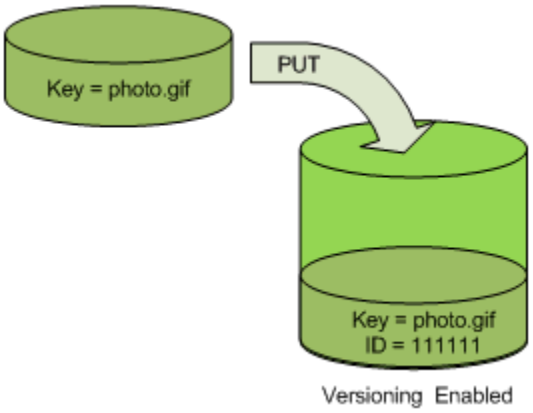
トピック

- バージョニングが有効なバケットへのオブジェクトの追加
- バージョニングが有効なバケットでのオブジェクトのリスト取得
- バージョニングが有効なバケットからのオブジェクトバージョンの取得
- バージョニングが有効なバケットからのオブジェクトバージョンの削除
- バージョン管理されたオブジェクトのアクセス許可の設定

バージョニングが有効なバケットへのオブジェクトの追加

バケットでバージョニングを有効にすると、Amazon S3 は (PUT、POST、または CopyObject を使用して) バケットに保存されたすべてのオブジェクトに、一意のバージョン ID を自動的に追加します。

次の図は、バージョニングが有効なバケットにオブジェクトが追加されたときに、Amazon S3 がそのオブジェクトに一意のバージョン ID を追加する方法を示しています。



> **Note**
>
> Amazon S3 が割り当てるバージョン ID の値は URL セーフです (URI の一部として含めることができます) 。

バージョニングの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。バージョニングが有効なバケットにオブジェクトバージョンを追加するときは、コンソール、AWS SDK、REST API を使用します。

コンソールを使用する

手順については、オブジェクトのアップロード を参照してください。

AWS SDK の使用

AWS SDK for Java、.NET、PHP を使用してオブジェクトをアップロードする例については、「オブジェクトのアップロード」を参照してください。バージョニングが設定されていないバケットでも、バージョニングが有効なバケットでもオブジェクトをアップロードする例は同じです。ただし、バージョニングが有効なバケットでは、Amazon S3 によってバージョン番号が割り当てられます。それ以外の場合、バージョン番号は null になります。

他の AWS SDK の使用の詳細については、「AWS デベロッパーセンター」を参照してください。

REST API の使用

バージョニングが有効なバケットにオブジェクトを追加するには

1. PutBucketVersioning リクエストを使用して、バケットでバージョニングを有効にします。

   詳細については、「Amazon Simple Storage Service API リファレンス」の「PutBucketVersioning」を参照してください。
2. PUT、POST、または CopyObject リクエストを送信して、バケットにオブジェクトを格納します。

バージョニングが有効なバケットにオブジェクトを追加すると、Amazon S3 は、次の例に示すとおり、x-amz-version-id レスポンスヘッダーでオブジェクトのバージョン ID を返します。

```
x-amz-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY
```

バージョニングが有効なバケットでのオブジェクトのリスト取得

このセクションでは、バージョニングが有効なバケットの、オブジェクトバージョンをリスト化する例を説明します。Amazon S3 では、オブジェクトのバージョン情報は、バケットに関連付けられているバージョンのサブリソースに格納されます。詳細については、「バケット設定オプション」を参照してください。バージョニングが有効なバケット内のオブジェクトを一覧表示するには、ListBucketVersions アクセス許可が必要です。

S3 コンソールの使用

以下の手順では、Amazon S3 コンソールを使用して、オブジェクトの異なるバージョンを表示する方法を説明します。

オブジェクトの複数のバージョンを表示する方法

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets] (バケット) リストで、オブジェクトが含まれるバケットの名前を選択します。
3. バケット内のオブジェクトのバージョン一覧を表示するには、[Show versions] (バージョンのリスト) スイッチを選択します。

   各オブジェクトバージョンについて、一意のバージョン ID、そのバージョンが作成された日時、その他のプロパティがコンソールに表示されます。(バージョニング状態を設定する前にバケットに格納されているオブジェクトには、バージョン ID [null] が付けられています)。

   バージョンのないオブジェクトをリストするには、[バージョンのリスト] スイッチを選択します。

オブジェクトのバージョンは、コンソールの [オブジェクト概要] ペインでも、確認、ダウンロード、削除が行えます。詳細については、「Amazon S3 コンソールでのオブジェクトの概要の表示」を参照してください。

> **Note**
>
> 300 バージョンより前のオブジェクトバージョンにアクセスするには、AWS CLI またはオブジェクトの URL を使用する必要があります。

> **Important**
>
> 最新 (現在) のバージョンとして削除された場合のみ、オブジェクトを復元できます。削除されたオブジェクトの以前のバージョンを復元することはできません。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

AWS SDK の使用

このセクションの例では、バージョニングが有効なバケットのオブジェクトを一覧表示する方法を示します。各リクエストは最大 1,000 個のバージョンを返します (これ未満の数を指定した場合を除きます)。バケット内のバージョン数が、この上限数を超えている場合は、複数のリクエストを送

信してすべてのバージョンを一覧表示します。このように「ページ」に分けて結果を返すプロセスは、ページ分割と呼ばれます。

ページ分割の仕組みを示すために、例ではレスポンスごとのオブジェクトバージョン数を 2 個に制限しています。各例では、最初のページの結果を取得すると、バージョンリストが切り詰められているかどうかを確認します。切り詰められている場合は、引き続きページの取得を繰り返し、すべてのバージョンを取得します。

> **Note**
>
> 以下の例は、バージョニングが有効になっていないバケットや、個別のバージョンを持たないオブジェクトにも使用できます。これらの場合、Amazon S3 で返される各オブジェクトのバージョン ID は null になります。

他の AWS SDK の使用の詳細については、「AWS デベロッパーセンター」を参照してください。

Java

作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListVersionsRequest;
import com.amazonaws.services.s3.model.S3VersionSummary;
import com.amazonaws.services.s3.model.VersionListing;

public class ListKeysVersioningEnabledBucket {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
```

```
                    .withRegion(clientRegion)
                    .build();

            // Retrieve the list of versions. If the bucket contains more versions
            // than the specified maximum number of results, Amazon S3 returns
            // one page of results per request.
            ListVersionsRequest request = new ListVersionsRequest()
                    .withBucketName(bucketName)
                    .withMaxResults(2);
            VersionListing versionListing = s3Client.listVersions(request);
            int numVersions = 0, numPages = 0;
            while (true) {
                numPages++;
                for (S3VersionSummary objectSummary :
 versionListing.getVersionSummaries()) {
                    System.out.printf("Retrieved object %s, version %s\n",
                            objectSummary.getKey(),
                            objectSummary.getVersionId());
                    numVersions++;
                }
                // Check whether there are more pages of versions to retrieve. If
                // there are, retrieve them. Otherwise, exit the loop.
                if (versionListing.isTruncated()) {
                    versionListing =
 s3Client.listNextBatchOfVersions(versionListing);
                } else {
                    break;
                }
            }
            System.out.println(numVersions + " object versions retrieved in " +
 numPages + " pages");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ListObjectsVersioningEnabledBucketTest
    {
        static string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main(string[] args)
        {
            s3Client = new AmazonS3Client(bucketRegion);
            GetObjectListWithAllVersionsAsync().Wait();
        }

        static async Task GetObjectListWithAllVersionsAsync()
        {
            try
            {
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    // You can optionally specify key name prefix in the request
                    // if you want list of object versions of a specific object.

                    // For this example we limit response to return list of 2
 versions.
                    MaxKeys = 2
                };
                do
                {
```

```
                    ListVersionsResponse response = await
 s3Client.ListVersionsAsync(request);
                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }

                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
                    else
                    {
                        request = null;
                    }
                } while (request != null);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

REST API の使用

Example - バケット内のすべてのオブジェクトバージョンをリスト化

バケット内の各オブジェクトのすべてのバージョンを一覧表示するには、versions リクエストで GET Bucket サブリソースを使用します。Amazon S3 で取得できるオブジェクトは最大 1,000 個

です。各オブジェクトバージョンが 1 個のオブジェクトとしてフルにカウントされます。したがって、バケット内に 2 つのキー (例: photo.gif と picture.jpg) があり、最初のキーに 990 個のバージョン、2 番目のキーに 400 個のバージョンがある場合、1 つのリクエストで取得されるバージョンは、photo.gif の 990 個と、picture.jpg のうち最新の 10 個のみです。

Amazon S3 は、最も新しく保存されたバージョンから開始して、保存された順序でオブジェクトバージョンを返します。

GET Bucket リクエストに、versions サブリソースを含めます。

```
GET /?versions HTTP/1.1
Host: bucketName.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 +0000
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

Example - キーのすべてのバージョンの取得

オブジェクトバージョンのサブセットを取得するには、GET Bucket のリクエストパラメータを使用します。詳細については、「GET Bucket」を参照してください。

1. prefix パラメータを、取得したいオブジェクトのキーに設定します。
2. GET Bucket サブリソースおよび versions を使用して、prefix リクエストを送信します。

   ```
   GET /?versions&prefix=objectName HTTP/1.1
   ```

Example - プレフィックスを使用したオブジェクトの取得

次の例では、キーが myObject であるか、またはそれで始まるオブジェクトを取得します。

```
GET /?versions&prefix=myObject HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

オブジェクトのすべてのバージョンのサブセットを取得するには、他のリクエストパラメータを使用します。詳細については、「Amazon Simple Storage Service API リファレンス」の「GET Bucket」を参照してください。

Example - レスポンスが切り詰められた後の残ったオブジェクトのリスト取得

GET リクエストで返すことができるオブジェクトの数が max-keys の値を超えた場合、レスポンスには <isTruncated>true</isTruncated> と、リクエストを満たすが返されなかった最初のキー (NextKeyMarker 内) および最初のバージョン ID (NextVersionIdMarker 内) が含まれます。GET リクエストを満たす追加のオブジェクトを取得する後続のリクエストで、これらの戻り値を開始位置として使用します。

バケットの元の GET Bucket versions リクエストを満たす追加のオブジェクトを取得するには、以下の手順に従ってください。key-marker、version-id-marker、NextKeyMarker、および NextVersionIdMarker の詳細については、「Amazon Simple Storage Service API リファレンス」の「GET Bucket」を参照してください。

以下は、元の GET リクエストを満たす追加のレスポンスです。

- key-marker の値を、前のレスポンスの NextKeyMarker で返されたキーに設定します。
- version-id-marker の値を、前のレスポンスの NextVersionIdMarker で返されたバージョン ID に設定します。
- GET Bucket versions および key-marker を使用して version-id-marker リクエストを送信します。

Example - 指定したキーおよびバージョン ID を始点としたオブジェクトの取得

```
GET /?versions&key-marker=myObject&version-id-marker=298459348571 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

AWS CLI の使用

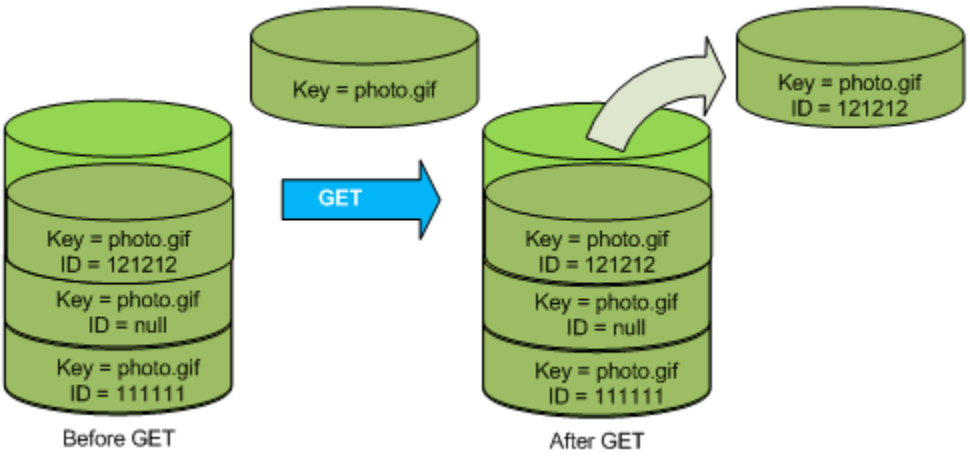
次のコマンドは、バケット内のオブジェクトのすべてのバージョンに関するメタデータを返します。

```
aws s3api list-object-versions --bucket example-s3-bucket1
```

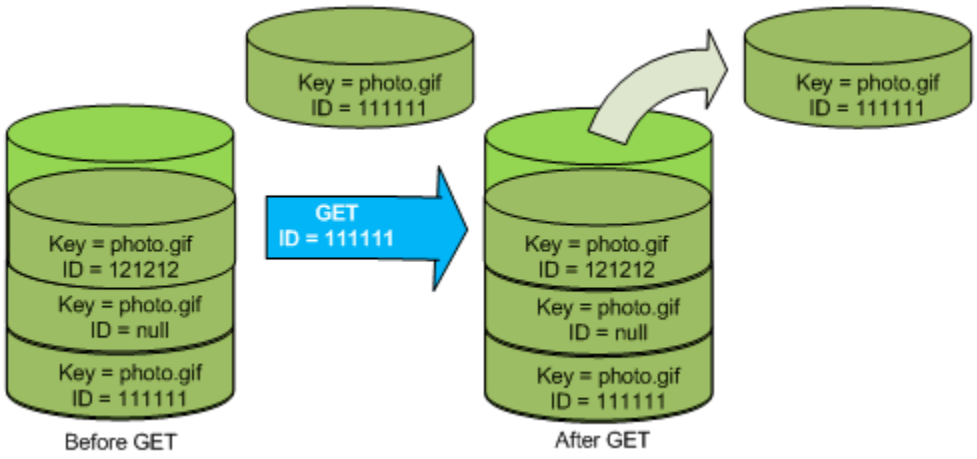
list-object-versions の使用に関する詳細は、「AWS CLI コマンドリファレンス」の「list-object-versions」を参照してください。

バージョニングが有効なバケットからのオブジェクトバージョンの取得

Amazon S3 のバージョニングとは、同じバケット内に、オブジェクトの複数のバリアントを保持する手段のことです。シンプルな GET リクエストは、オブジェクトの最新バージョンを取得します。次の図は、GET がオブジェクト (photo.gif) の最新バージョンを返す方法を示しています。



特定のバージョンを取得するには、そのバージョン ID を指定する必要があります。次の図は、GET versionId リクエストがオブジェクトの指定したバージョン (最新とは限らない) を取得する方法を示しています。



コンソール、AWS SDK、または REST API を使用して、Amazon S3 にオブジェクトバージョンを取得できます。

> **Note**
>
> 300 バージョンより前のオブジェクトバージョンにアクセスするには、AWS CLI またはオブジェクトの URL を使用する必要があります。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets] (バケット) リストで、オブジェクトが含まれるバケットの名前を選択します。
3. オブジェクト一覧から、オブジェクトの名前を選択します。
4. [バージョン] を選択します。

   Amazon S3 にオブジェクトのバージョンがすべて表示されます。
5. 取得するバージョンのバージョン ID の、横のチェックボックスをオンにします。
6. [アクション] を選択し、[ダウンロード] を選択して、オブジェクトを保存します。

オブジェクトの概要パネルでも、オブジェクトのバージョンの確認、ダウンロード、および削除ができます。詳細については、「Amazon S3 コンソールでのオブジェクトの概要の表示」を参照してください。

> **Important**
>
> 最新 (現在) のバージョンとして削除された場合のみ、オブジェクトを復元できます。削除されたオブジェクトの以前のバージョンを復元することはできません。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

AWS SDK の使用

バージョニングが無効なバケットと有効なバケットにオブジェクトをアップロードする場合の例は、同一です。ただし、バージョニングが有効なバケットには、Amazon S3 がバージョン番号を割り当てます。それ以外の場合、バージョン番号は null になります。

AWS SDKs for Java、.NET、および PHP を使用してオブジェクトをダウンロードする例については、オブジェクトのダウンロードを参照してください。

.NET と Rust のAWS SDK を使用してオブジェクトのバージョンを一覧表示する例については、「Amazon S3 バケット内のオブジェクトのバージョンを一覧表示する」を参照してください。

REST API の使用

特定のオブジェクトバージョンを取得するには

1. versionId を、取得するオブジェクトのバージョン ID に設定します。
2. GET Object versionId リクエストを送信します。

Example - バージョニングされたオブジェクトの取得

次のリクエストは、L4kqtJlcpXroDTDmpUMLUo のバージョン my-image.jpg を取得します。

```
GET /my-image.jpg?versionId=L4kqtJlcpXroDTDmpUMLUo HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

オブジェクトの (コンテンツではなく) メタデータのみを取得できます。詳細については、「the section called “バージョンメタデータの取得”」を参照してください。

以前のオブジェクトバージョンを復元する方法については、「the section called “以前のバージョンの復元”」を参照してください。

オブジェクトバージョンのメタデータの取得

オブジェクトのメタデータのみを取得するには (コンテンツを除く)、HEAD オペレーションを使用します。デフォルトでは、最新バージョンのメタデータが取得されます。特定のオブジェクトバージョンのメタデータを取得するには、そのバージョン ID を指定します。

オブジェクトバージョンのメタデータを取得するには

1. versionId を、メタデータを取得するオブジェクトのバージョン ID に設定します。
2. HEAD Object versionId リクエストを送信します。

Example - バージョニングされたオブジェクトのメタデータの取得

次のリクエストは、my-image.jpg のバージョン 3HL4kqCxf3vjVBH40Nrjfkd のメタデータを取得します。

```
HEAD /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

以下に、サンプルレスポンスを示します。

```
HTTP/1.1 200 OK
x-amz-id-2: ef8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC143432E5
x-amz-version-id: 3HL4kqtJlcpXroDTDmjVBH40Nrjfkd
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: AmazonS3
```

以前のバージョンの復元

バージョニングを使用すると、オブジェクトの以前のバージョンを取得できます。この機能を実行するには 2 つの方法があります。

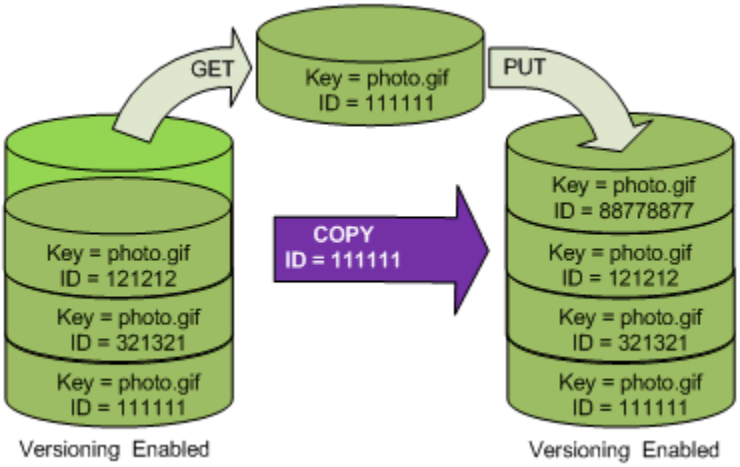
- オブジェクトの以前のバージョンを同じバケットにコピーします。

  コピーされたオブジェクトはそのオブジェクトの最新バージョンになり、すべてのオブジェクトバージョンが維持されます。
- オブジェクトの最新バージョンを完全に削除します。

  最新のオブジェクトバージョンを削除すると、結果として、以前のバージョンがそのオブジェクトの最新バージョンになります。

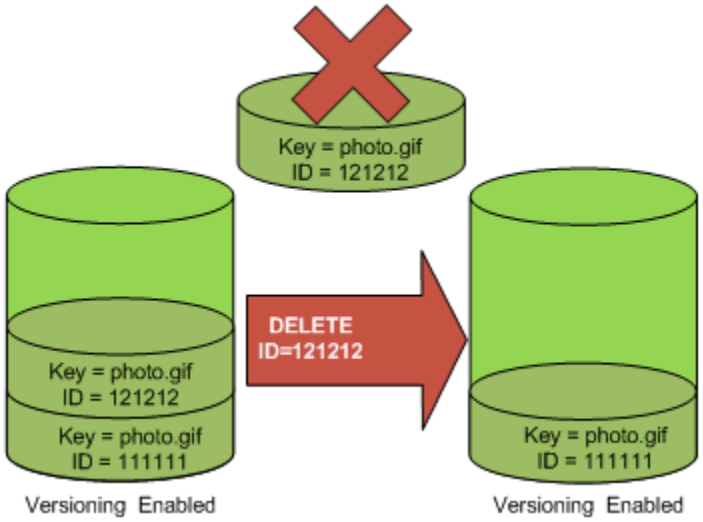
すべてのオブジェクトバージョンが維持されるため、オブジェクトの特定のバージョンをバケットにコピーすることにより、以前の任意のバージョンを最新バージョンにすることができます。次の図では、ソースオブジェクト (ID = 111111) が同じバケットにコピーされます。Amazon S3 が新しい ID (88778877) を指定し、それがオブジェクトの最新バージョンになります。したがって、バケットには元のオブジェクトバージョン (111111) とそのコピー (88778877) の両方が存在します。以前のバージョンを取得してからアップロードして最新のバージョンにする方法の詳細については、バー

ジョニングが有効なバケットからのオブジェクトバージョンの取得およびオブジェクトのアップロードを参照してください。



後続の GET が、バージョン 88778877 を取得します。

次の図では、オブジェクトの最新バージョン (121212) を削除して、以前のバージョン (111111) を最新のオブジェクトとして残す方法を示しています。オブジェクトの削除の詳細については、「1 つのオブジェクトの削除」を参照してください。



後続の GET が、バージョン 111111 を取得します。

> **Note**
>
> オブジェクトバージョンをバッチで復元する場合は、CopyObject オペレーションを使用できます。CopyObject オペレーションは、マニフェストで指定した各オブジェクトをコピーします。ただし、オブジェクトは、必ずしもマニフェストに表示されている順序と同じ順序

でコピーされるとは限らないため、注意してください。バージョン管理されたバケットでは、現在のバージョンまたは以前のバージョンの順序を維持することが重要な場合は、最初に以前のバージョンをすべてコピーする必要があります。次に、最初のジョブが完了したら、後続のジョブで現在のバージョンをコピーします。

以前のオブジェクトバージョンを復元する

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets] (バケット) リストで、オブジェクトが含まれるバケットの名前を選択します。
3. オブジェクト一覧から、オブジェクトの名前を選択します。
4. [バージョン] を選択します。

   Amazon S3 にオブジェクトのバージョンがすべて表示されます。
5. 取得するバージョンのバージョン ID の、横のチェックボックスをオンにします。
6. [アクション] を選択し、[ダウンロード] を選択して、オブジェクトを保存します。

オブジェクトの概要パネルでも、オブジェクトのバージョンの確認、ダウンロード、および削除ができます。詳細については、「Amazon S3 コンソールでのオブジェクトの概要の表示」を参照してください。

> **Important**
>
> 最新 (現在) のバージョンとして削除された場合のみ、オブジェクトを復元できます。削除されたオブジェクトの以前のバージョンを復元することはできません。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

AWS SDK の使用

他の AWS SDK の使用の詳細については、「AWS デベロッパーセンター」を参照してください。

Python

以下の Python コード例は、指定されたロールバックバージョンの後で生成されたすべてのバージョンを削除することによって、バージョン管理されたオブジェクトの以前のバージョンを復元します。

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
```

```
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

バージョニングが有効なバケットからのオブジェクトバージョンの削除

オブジェクトのバージョンは Amazon S3 バケットからいつでも削除できます。さらに、明確なライフサイクルが定義されているオブジェクトのライフサイクル設定ルールを定義することにより、最新のオブジェクトバージョンを有効期限切れにしたり、以前のオブジェクトバージョンを完全に削除したりするよう Amazon S3 にリクエストできます。バケットでバージョニングが有効であるか、バージョニングが停止されている場合、ライフサイクル設定アクションは次のように動作します。

- `Expiration` アクションは、最新のオブジェクトバージョンに適用されます。Amazon S3 は、最新のオブジェクトバージョンを削除するのではなく、削除マーカーを追加してそれを最新バージョンにします。これにより、最新バージョンは以前のバージョンとして保持されます。
- `NoncurrentVersionExpiration` アクションは、以前のオブジェクトバージョンに適用されます。Amazon S3 はこれらのオブジェクトバージョンを完全に削除します。完全に削除したオブジェクトは復元できません。

S3 ライフサイクルの詳細については、「ストレージのライフサイクルの管理」および「S3 ライフサイクル設定の例」を参照してください。

バケットにある現在のオブジェクトバージョンと最新でないオブジェクトバージョンの数を確認するには、Amazon S3 ストレージレンズメトリクスを使用できます。S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージの分析機能です。詳細については、「S3 Storage Lens を使用したストレージコストの最適化」を参照してください。メトリクスの完全なリストについては、「S3 ストレージレンズメトリクスに関する用語集」を参照してください。

> **Note**
>
> 通常の Amazon S3 料金は、最新ではないオブジェクトのバージョンなど、保存または移行されるオブジェクトのバージョンごとに適用されます。詳細については、「Amazon S3 の料金」を参照してください。

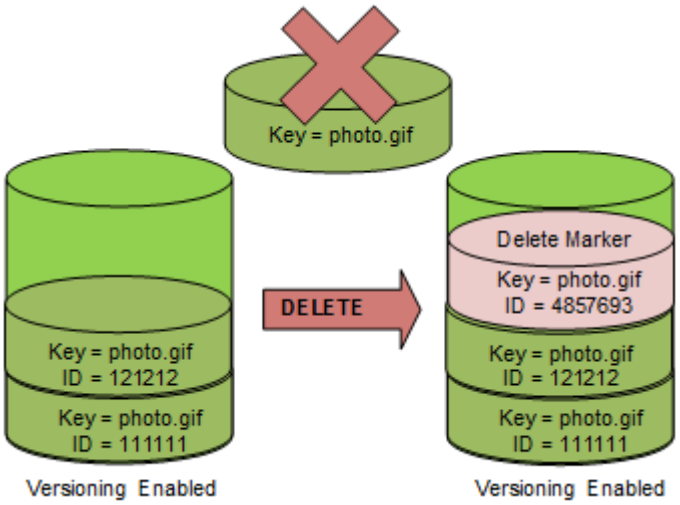
リクエストのユースケースを削除する

DELETE リクエストには次のユースケースがあります。

- バージョニングが有効になっている場合、単純な DELETE はオブジェクトを完全に削除することはできません (単純な DELETE リクエストとは、バージョン ID を指定しないリクエストです)。代わりに、Amazon S3 はバケットに削除マーカーを挿入します。このマーカーが新しい ID を持つオブジェクトの最新バージョンになります。

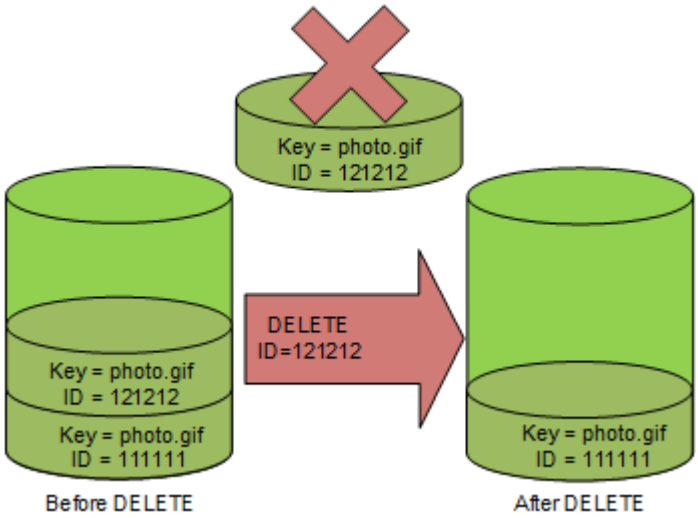
  最新バージョンが削除マーカーであるオブジェクトを GET しようとすると、Amazon S3 は、オブジェクトが (消去されていなくても) 削除されたものとして動作し、エラー 404 を返します。詳細については、「削除マーカーの使用」を参照してください。

  次の図は、シンプルな DELETE が、指定したオブジェクトを実際には削除しないことを示しています。代わりに、Amazon S3 は削除マーカーを挿入します。



- バージョニングされたオブジェクトを完全に削除するときは、必ず DELETE Object versionId を使用します。

  次の図は、指定したオブジェクトバージョンを削除することによって、そのオブジェクトを完全に削除する方法を示しています。



オブジェクトバージョンを削除するには

Amazon S3 内のオブジェクトバージョンは、コンソール、AWS SDK、REST API、または AWS Command Line Interface を使用して削除できます。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets] (バケット) リストで、オブジェクトが含まれるバケットの名前を選択します。
3. オブジェクト一覧から、オブジェクトの名前を選択します。
4. [バージョン] を選択します。

   Amazon S3 にオブジェクトのバージョンがすべて表示されます。
5. 完全に削除するバージョンのバージョン ID の横のチェックボックスをオンにします。
6. [削除] を選択します。
7. [オブジェクトを完全に削除しますか?] に、**`permanently delete`** と入力します。

   > ⚠ Warning
   >
   > オブジェクトバージョンを完全に削除すると、アクションを元に戻すことができません。

8. [オブジェクトの削除] を選択します。

Amazon S3 がオブジェクトのバージョンを削除します。

AWS SDK の使用

AWS SDK for Java、.NET、PHP を使用してオブジェクトを削除する例については、Amazon S3 オブジェクトの削除 を参照してください。バージョニングが無効なバケットと有効なバケットでオブジェクトを削除する場合の例は、同一です。ただし、バージョニングが有効なバケットには、Amazon S3 がバージョン番号を割り当てます。それ以外の場合、バージョン番号は null になります。

他の AWS SDK の使用の詳細については、「AWS デベロッパーセンター」を参照してください。

Python

次の Python コード例は、すべてのバージョンを削除することによってバージョニングされたオブジェクトを完全に削除する方法を示しています。

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

REST API の使用

オブジェクトの特定のバージョンを削除するには

- DELETE でバージョン ID を指定します。

Example - 特定のバージョンの削除

次の例では、photo.gif のバージョン UIORUnfnd89493jJFJ を削除します。

```
DELETE /photo.gif?versionId=UIORUnfnd89493jJFJ HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Type: text/plain
Content-Length: 0
```

AWS CLI の使用

次のコマンドは、*example-s3-bucket1* という名前のバケットから test.txt という名前のオブジェクトを削除します。特定のバージョンのオブジェクトを削除するには、バケット所有者として、バージョン ID サブリソースを使用する必要があります。

```
aws s3api delete-object --bucket example-s3-bucket1 --key test.txt --version-
id versionID
```

delete-object に関する詳細については、「AWS CLI コマンドリファレンス」の「delete-object」を参照してください。

オブジェクトバージョンの削除の詳細については、以下のトピックを参照してください。

- 削除マーカーの使用
- 古いバージョンを現行バージョンにするための削除マーカーの削除
- MFA Delete が有効なバケットからのオブジェクトの削除

削除マーカーの使用

Amazon S3 の削除マーカーは、単純な DELETE リクエストで指定された、バージョニングされたオブジェクトのプレースホルダー (またはマーカー) です。単純な DELETE リクエストとは、バージョン ID を指定しないリクエストです。オブジェクトがバージョニングが有効なバケット内にあ

るため、そのオブジェクトは削除されません。ただし、削除マーカーにより、Amazon S3 はオブジェクトが削除されたかのように動作します。削除マーカーに対しては、Amazon S3 API DELETE コールを使用できます。これを行うには、適切なアクセス許可を持つ AWS Identity and Access Management (IAM) ユーザーまたはロールを使用して DELETE リクエストを行う必要があります。

削除マーカーには、他のすべてのオブジェクトと同様に、キー名 (またはキー) とバージョン ID があります。ただし、削除マーカーは次のような点で他のオブジェクトとは異なります。

- 削除マーカーには、データが関連付けられていません。
- 削除マーカーには、アクセスコントロールリスト (ACL) の値が関連付けられていません。
- 削除マーカーに対する GET リクエストを発行しても、削除マーカーにはデータがないため、GET リクエストでは何も取得されません。具体的には、GET リクエストで versionId を指定しない場合、404 (Not Found) エラーが表示されます。

削除マーカーにより、Amazon S3 内のストレージに対して最低料金が発生します。削除マーカーのストレージサイズは、その削除マーカーのキー名のサイズと同じです。キー名は一連の Unicode 文字です。キー名の UTF-8 エンコーディングでは、名前の文字ごとに 1～4 バイトのストレージがバケットに追加されます。削除マーカーは S3 Standard ストレージクラスに保存されます。

削除マーカーの保有数と、保存先のストレージクラスを確認する場合は、Amazon S3 ストレージレンズを使用できます。詳細については、Amazon S3 ストレージレンズを使用してストレージのアクティビティと使用状況を評価するおよびAmazon S3 Storage Lens のメトリクスに関する用語集を参照してください。

有効なキー名の詳細については、オブジェクトキー名の作成 を参照してください。削除マーカーの削除の詳細については、「削除マーカーの管理」を参照してください。

Amazon S3 のみが削除マーカーを作成することができ、バージョニングが有効なバケットまたは停止されたバケット内のオブジェクトに対して DeleteObject リクエストが送信されるたびにこの作成を行います。DELETE リクエストで指定したオブジェクトは実際には削除されず、代わりに削除マーカーがオブジェクトの最新バージョンになります。オブジェクトのキー名 (またはキー) が削除マーカーのキーになります。

リクエストで versionId を指定せずにオブジェクトを取得するときに、現在のバージョンが削除マーカーである場合、Amazon S3 は次のように応答します。

- 404 (Not Found) エラー
- レスポンスヘッダー、x-amz-delete-marker: true

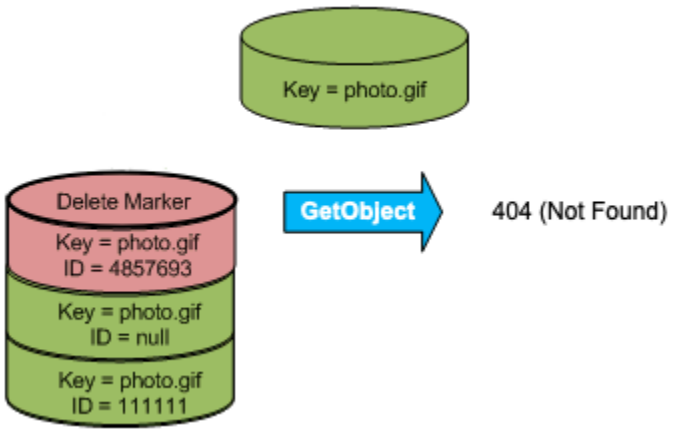
リクエストで versionId を指定してオブジェクトを取得するときに、指定したバージョンが削除マーカーである場合、Amazon S3 は次のように応答します。

- 405 (Method Not Allowed) エラー
- レスポンスヘッダー、x-amz-delete-marker: true
- レスポンスヘッダー、Last-Modified: timestamp (HeadObject または GetObject API オペレーションを使用する場合のみ)

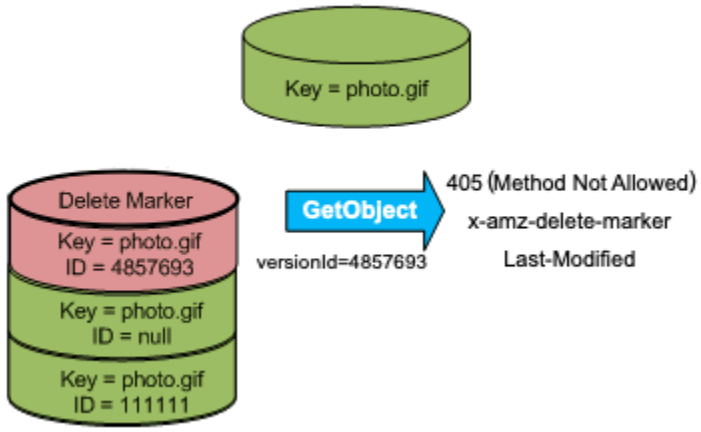
x-amz-delete-marker: true レスポンスヘッダーから、アクセスしたオブジェクトが削除マーカーであったことがわかります。値が false である場合、オブジェクトの現在のバージョンまたは指定したバージョンは削除マーカーではないため、このレスポンスヘッダーは false を返しません。

Last-Modified レスポンスヘッダーには削除マーカーの作成時間が表示されます。

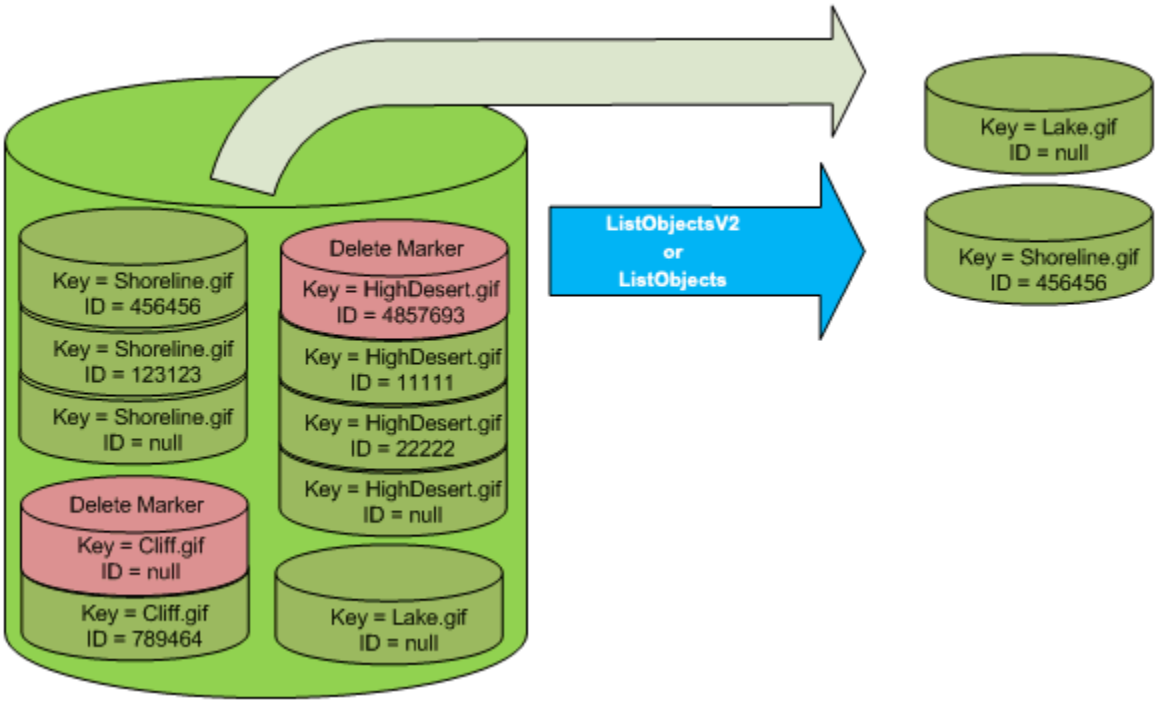
次の図は、現在のバージョンが削除マーカーであるオブジェクトに対する GetObject API コールから返される 404 (No Found) エラーと、レスポンスヘッダーに含まれる x-amz-delete-marker: true を示しています。



リクエストで versionId を指定してオブジェクトに対して GetObject コールを行った場合、指定したバージョンが削除マーカーであると、Amazon S3 は 405 (Method Not Allowed) エラーで応答し、レスポンスヘッダーには x-amz-delete-marker: true と Last-Modified: timestamp が含まれます。



削除マーカー (およびオブジェクトのその他のバージョン) をリストするには、ListObjectVersions リクエストで `versions` サブリソースを使用するしかありません。次の図で、ListObjectsV2 リクエストや ListObjects リクエストは、現在のバージョンが削除マーカーであるオブジェクトを返さないことを示しています。



削除マーカーの管理

期限切れの削除マーカーを自動的にクリーンアップするライフサイクルの設定

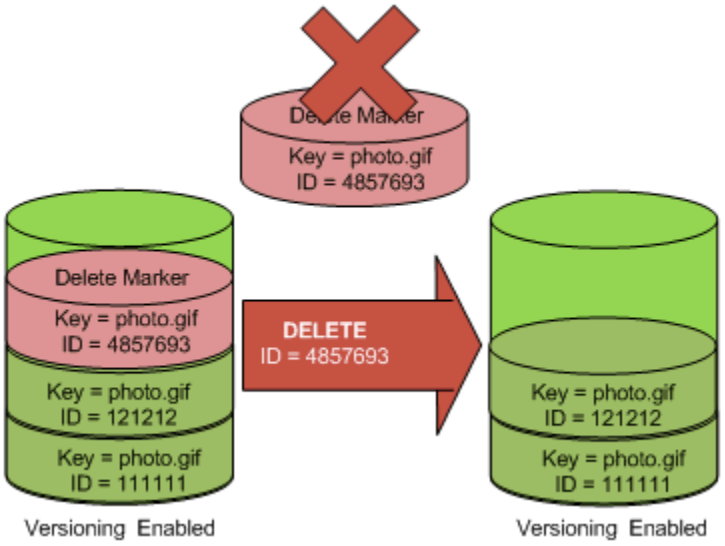
期限切れのオブジェクト削除マーカーは、すべてのオブジェクトバージョンが削除され、単一の削除マーカーだけが残っている場合のマーカーです。ライフサイクル設定が現行バージョンを削除するように設定されている、または `ExpiredObjectDeleteMarker` アクションが明示的に設定されてい

る場合、Amazon S3 は期限切れのオブジェクト削除マーカーを削除します。例については、「例 7: 期限切れオブジェクト削除マーカーを削除する」を参照してください。

古いバージョンを現行バージョンにするための削除マーカーの削除

バージョニングが有効なバケット内のオブジェクトを削除すると、すべてのバージョンがバケット内に残り、Amazon S3 はオブジェクトの削除マーカーを作成します。削除したオブジェクトを復元するには、この削除マーカーを削除する必要があります。バージョニングと削除マーカーの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

削除マーカーを完全に削除するには、DeleteObject versionId リクエストにそのバージョン ID を含める必要があります。次の図は、DeleteObject versionId リクエストが削除マーカーを完全に削除する方法を示しています。

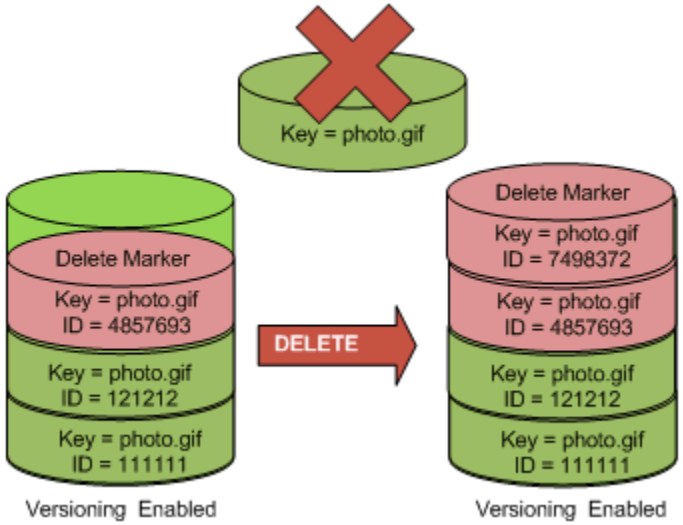


削除マーカーを削除すると、シンプルな GET リクエストがオブジェクトの現行バージョン ID (121212) を取得するようになります。

> **Note**
>
> 現行バージョンが削除マーカーである場合に、削除マーカーのバージョン ID を指定せずに DeleteObject リクエストを使用すると、Amazon S3 は削除マーカーを削除せず、その代わりに別の削除マーカーを PUTs します。

NULL バージョン ID を持つ削除マーカーを削除するには、DeleteObject リクエストのバージョン ID として NULL を渡す必要があります。以下の図は、現行バージョンが削除マーカーになっている

ときに、バージョンID なしで実行されたシンプルな DeleteObject リクエストが何も削除せず、その代わりに一意のバージョン ID (7498372) を持つ削除マーカーをさらに追加する状況を示しています。



S3 コンソールの使用

S3 バケットから削除されたフォルダ以外のオブジェクト (フォルダ内のオブジェクトも含む) を復元するには、以下の手順を実行します。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets (バケット)] リストで、目的のバケットの名前を選択します。
3. バケット内のオブジェクトのバージョン一覧を表示するには、[バージョンのリスト] を選択します。削除されたオブジェクトの削除マーカーを表示できます。
4. 削除したオブジェクトを復元するには、削除マーカーを削除する必要があります。復元するオブジェクトの削除マーカーの横にあるチェックボックスを選択し、次に [削除] を選択します。
5. [オブジェクトの削除] ページで削除を確認します。
   a. [オブジェクトを完全に削除しますか?] に、**`permanently delete`** と入力します。
   b. [オブジェクトの削除] を選択します。

> **Note**
>
> Amazon S3 コンソールでは、削除されたフォルダを元に戻すことはできません。AWS CLI または SDK を使用する必要があります。例については、AWS ナレッジセンターのバージョ

ニングが有効なバケットで削除された Amazon S3 オブジェクトを取得する方法を教えてください」を参照してください。

REST API の使用

削除マーカーを完全に削除するには

1. versionId を、削除する削除マーカーのバージョン ID に設定します。
2. DELETE Object versionId リクエストを送信します。

Example - 削除マーカーの削除

次の例では、photo.gif のバージョン 4857693 の削除マーカーを削除します。

```
DELETE /photo.gif?versionId=4857693 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

削除マーカーを削除すると、Amazon S3 はレスポンスに以下を含めます。

```
204 NoContent
x-amz-version-id: versionID
x-amz-delete-marker: true
```

AWS SDK の使用

他の AWS SDK の使用の詳細については、「AWS デベロッパーセンター」を参照してください。

Python

次の Python コード例は、オブジェクトから削除マーカーを削除し、最新でないバージョン (最新に最も近いもの) をオブジェクトの最新のバージョンにする方法を示しています。

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
```

```
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

MFA Delete が有効なバケットからのオブジェクトの削除

バケットのバージョニング設定で MFA Delete が有効になっている場合に、オブジェクトバージョンを完全に削除したり、バケットのバージョニング状態を変更したりするには、バケット所有者はリクエストに x-amz-mfa リクエストヘッダーを含める必要があります。x-amz-mfa を含むリクエストでは、HTTPS を使用する必要があります。

ヘッダーの値は、認証デバイスのシリアル番号、スペース、および認証デバイスに表示される認証コードの連結文字です。このリクエストヘッダーを含めないと、リクエストは失敗します。

認証デバイスの詳細については、「多要素認証」を参照してください。

Example - MFA Delete が有効なバケットからのオブジェクトの削除

次の例は、my-image.jpg を (バージョンを指定して) 削除しています。これは、MFA Delete が有効に設定されたバケットの中にあります。

[SerialNumber] と *[AuthenticationCode]* の間のスペースに注意してください。詳細については、「Amazon Simple Storage Service API リファレンス」の「DeleteObject」を参照してください。

```
DELETE /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTPS/1.1
Host: bucketName.s3.amazonaws.com
x-amz-mfa: 20899872 301749
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

MFA Delete の有効化の詳細については、「MFA 削除の設定」を参照してください。

バージョン管理されたオブジェクトのアクセス許可の設定

Amazon S3 のオブジェクトのアクセス許可は、バージョンレベルで設定されています。各バージョンには個々のオブジェクト所有者がいます。この所有者は、オブジェクトバージョンを作成する AWS アカウントです。したがって、同じオブジェクトの異なるバージョンに対して、異なるアクセス許可を設定することができます。これを行うには、PUT Object versionId acl リクエストでアクセス権限を設定するオブジェクトのバージョン ID を指定する必要があります。ACL の使用の詳細と手順については、「Amazon S3 用 Identity and Access Management」を参照してください。

Example - オブジェクトバージョンに対するアクセス許可の設定

次のリクエストは、キー my-image.jpg、バージョンID 3HL4kqtJvjVBH40Nrjfkd に対して、被付与者 BucketOwner@amazon.com の許可を FULL_CONTROL に設定するものです。

```
PUT /my-image.jpg?acl&versionId=3HL4kqtJvjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
Content-Length: 124

<AccessControlPolicy>
  <Owner>
    <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>mtd@amazon.com</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>BucketOwner@amazon.com</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  </AccessControlPolicy>
```

同様に、特定のオブジェクトバージョンに対するアクセス許可を取得するには、GET Object versionId acl リクエストでそのバージョン ID を指定する必要があります。デフォルトでは GET Object acl はオブジェクトの最新バージョンに対するアクセス許可を返すため、バージョン ID を含める必要があります。

Example - 指定したオブジェクトバージョンに対するアクセス許可の取得

次の例では、Amazon S3 はキー my-image.jpg、バージョン ID DVBH40Nr8X8gUMLUo に対するアクセス許可を返します。

```
GET /my-image.jpg?versionId=DVBH40Nr8X8gUMLUo&acl HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU
```

詳細については、「Amazon Simple Storage Service API リファレンス」の「GetObjectAcl」を参照してください。

バージョニングが停止されたバケットのオブジェクトの操作

Amazon S3 では、バケット内で同じオブジェクトの新しいバージョンが生成されないようにするには、バージョニングを停止します。この作業は、バケット内のオブジェクトのバージョンが 1 つだけ必要な場合に行います。あるいは、複数のバージョンで料金を発生させたくない場合などです。

バージョニングを停止しても、バケットに含まれる既存のオブジェクトは変更されません。変更されるのは、Amazon S3 が今後のリクエストでオブジェクトを処理する方法です。このセクションのトピックでは、バージョニングが停止されたバケットでの、オブジェクトの追加、取得、削除など、さまざまなオペレーションについて説明します。

S3 バージョニングの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。オブジェクトバージョンの取得の詳細については、「バージョニングが有効なバケットからのオブジェクトバージョンの取得」を参照してください。

トピック

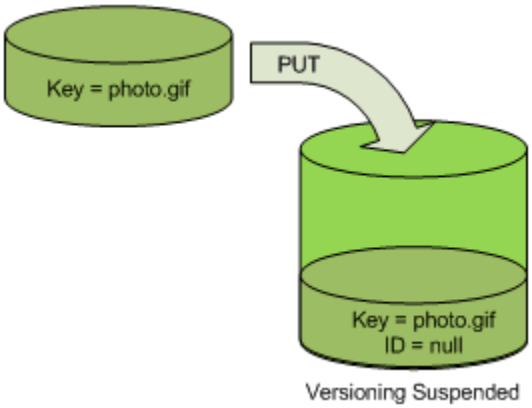
- バージョニングが停止されたバケットへのオブジェクトの追加
- バージョニングが停止されたバケットからのオブジェクトの取得
- バージョニングが停止されたバケットからのオブジェクトの削除

バージョニングが停止されたバケットへのオブジェクトの追加

Amazon S3 でバージョニングが停止されたバケットにオブジェクトを追加すると、null バージョン ID の付いたオブジェクトが作成されるか、または同じバージョン ID のオブジェクトバージョンが上書きされます。

バケットでバージョニングを停止すると、Amazon S3 は (PUT、POST、CopyObject を使用して) その後そのバケットに保存された後続のすべてのオブジェクトに、null バージョン ID を自動的に追加します。

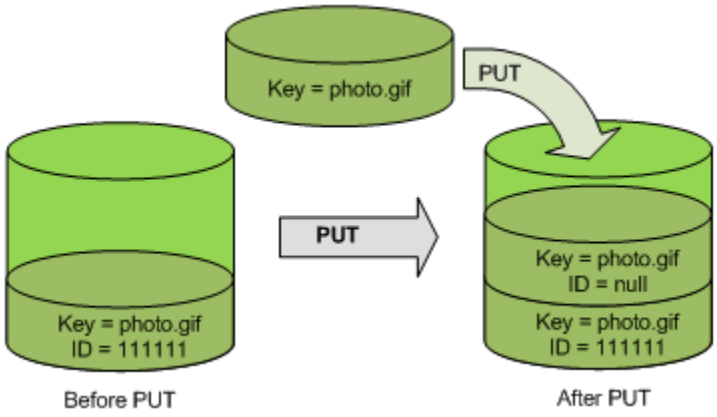
次の図は、バージョニングが停止されたバケットにオブジェクトが追加されたときに、Amazon S3 がそのオブジェクトにバージョン ID null を追加する方法を示しています。



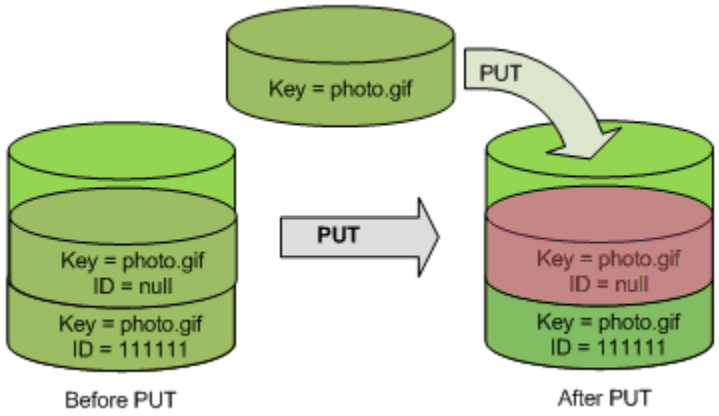
バケット内に null バージョンが既に存在しており、同じキーを持つ別のオブジェクトを追加した場合、元の null バージョンは、追加したオブジェクトによって上書きされます。

バケット内にバージョニングされたオブジェクトがある場合、PUT したバージョンはオブジェクトの最新バージョンになります。次の図は、バージョニングされたオブジェクトを含むバケットにオブジェクトを追加しても、バケット内に既に存在するオブジェクトを上書きしないことを示しています。

この場合、バケット内にバージョン 111111 がすでに存在しています。Amazon S3 は、追加されるオブジェクトに null のバージョン ID をアタッチし、バケット内に保存します。バージョン 111111 は上書きされません。



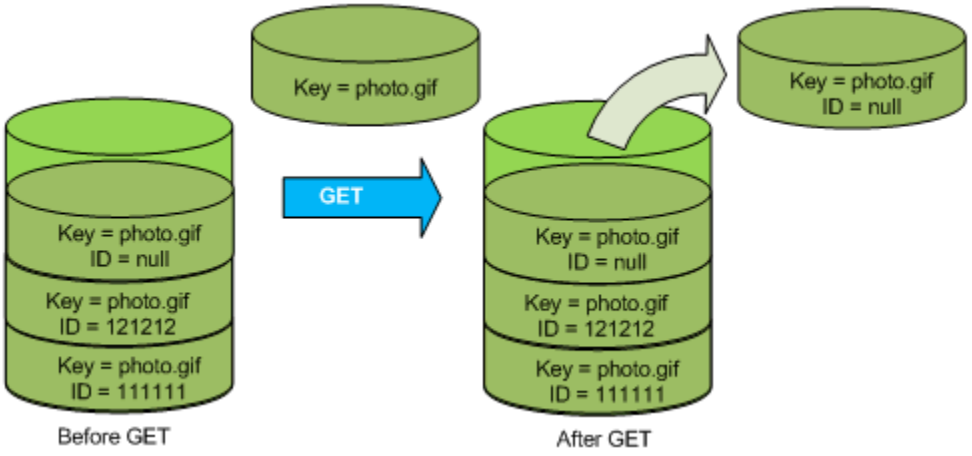
次の図に示すように、バケット内に null バージョンが既に存在する場合、null バージョンは上書きされます。



null バージョンのキーおよびバージョン ID (null) は PUT の前後で同じですが、バケット内に元々格納されていた null バージョンの内容は、バケット内のオブジェクト PUT の内容に置き換えられます。

バージョニングが停止されたバケットからのオブジェクトの取得

GET Object リクエストは、バケットでバージョニングを有効にしているかどうかにかかわらず、オブジェクトの最新バージョンを返します。次の図は、シンプルな GET がオブジェクトの最新バージョンを返す方法を示しています。



バージョニングが停止されたバケットからのオブジェクトの削除

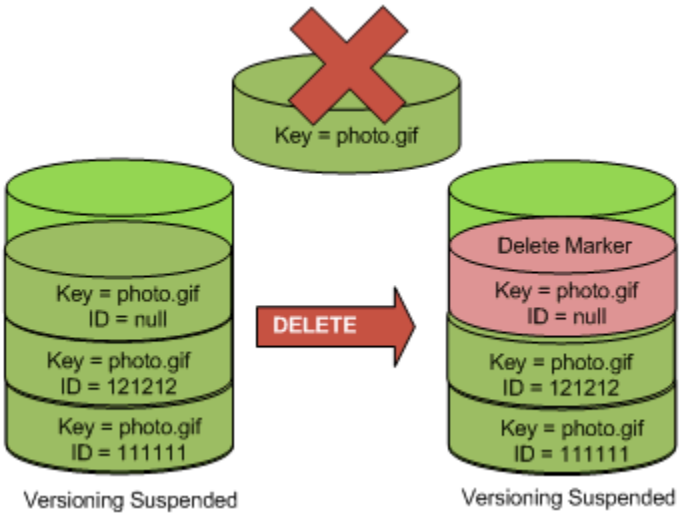
バージョニングが停止されたバケットからオブジェクトを削除することで、null バージョン ID の付いたオブジェクトを削除します。

バケットのバージョニングが停止されている場合、DELETE リクエストは次のように動作します。

- バージョン ID が null であるオブジェクトのみを削除できる。
- バケット内にオブジェクトの null バージョンが存在しない場合は、何も削除しません。

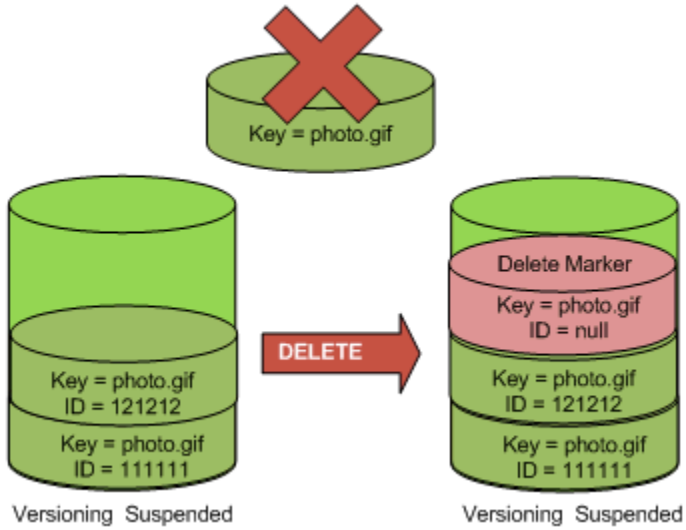
- バケットに削除マーカーを挿入する。

次の図は、単純な DELETE が null バージョンを削除する方法を示したものです (単純な DELETE リクエストとは、バージョン ID を指定しないリクエストです)。Amazon S3 は、その場所にバージョン ID null の削除マーカーを挿入します。



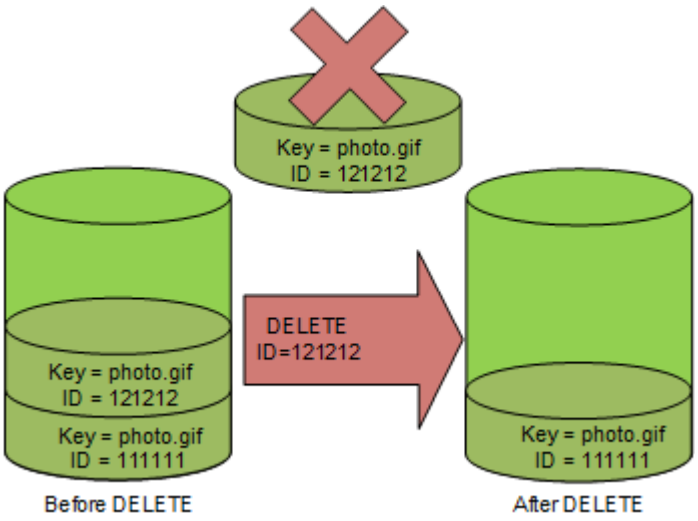
削除マーカーには内容がないため、削除マーカーに置き換えられるときに null バージョンの内容が失われることに注意してください。

次の図は、null バージョンが存在しないバケットを示しています。この場合、DELETE は何も削除せず、Amazon S3 は単に削除マーカーを挿入します。



バージョニングが停止されたバケットでも、バケット所有者は DELETE リクエスト内のバージョン IDを含む指定バージョンを完全に削除できます。次の図は、指定したオブジェクトバージョンを削除

することによって、そのオブジェクトのバージョンを完全に削除する方法を示しています。バケット所有者のみが、指定したオブジェクトバージョンを削除することができます。



Amazon S3 用の AWS Backup の使用

Amazon S3 は、AWS Backup とネイティブに統合されたフルマネージド型のポリシーベースのサービスであるため、バックアップポリシーを一元的に定義して、Amazon S3 内のデータを保護できます。バックアップポリシーを定義し、Amazon S3 リソースをポリシーに割り当てると、AWS Backup は Amazon S3 バックアップの作成を自動化し、バックアッププランで指定した暗号化されたバックアップボールトにバックアップを安全に保存します。

AWS Backup を Amazon S3 に使用すると、以下のアクションを実行できます。

- 継続的なバックアップと定期的なバックアップを作成します。継続的なバックアップは、特定の時点への復元の場合に便利であり、 および定期的なバックアップは、長期的なデータ-保持のニーズを満たすのに便利です。
- バックアップポリシーを一元的に構成して、バックアップのスケジュールと保存を自動化します。
- Amazon S3 データのバックアップを、指定した特定の時点に復元します。

AWS Backup と同様に、S3 バージョニングと S3 レプリケーションを使用して、偶発的な削除から回復し、独自のセルフリカバリーオペレーションを実行できます。

前提条件

AWS Backup がバケットをバックアップするには、その前に、バケットの S3 バージョニングを有効にする必要があります。

> **Note**
>
> バックアップするバージョニングが有効なバケットのライフサイクルの有効期限ルールを設定することをお勧めします。ライフサイクルの有効期限を設定しなかった場合、AWS Backup は Amazon S3 データのすべてのバージョンを保持するため、Amazon S3 のストレージコストが増加する可能性があります。

開始方法

Amazon S3 で AWS Backup の使用を開始するには、AWS Backup デベロッパーガイドの「Amazon S3 バックアップの作成」を参照してください。

制約と制限

制限について学ぶには、AWS Backup デベロッパーガイドの「Amazon S3 バックアップの作成」を参照してください。

アーカイブされたオブジェクトの操作

アクセス頻度の低いオブジェクトのストレージコストを削減するには、それらのオブジェクトをアーカイブできます。オブジェクトをアーカイブすると、そのオブジェクトは低コストのストレージに移動されるため、リアルタイムでアクセスすることはできません。

アーカイブされたオブジェクトにはリアルタイムではアクセスできませんが、ストレージクラスによっては数分または数時間で復元できます。アーカイブされたオブジェクトは、Amazon S3 コンソール、S3 バッチオペレーション、REST API、AWS SDK、および AWS Command Line Interface (AWS CLI) を使用して復元できます。手順については、アーカイブされたオブジェクトの復元 を参照してください。

次のストレージクラスまたは階層の Amazon S3 オブジェクトはアーカイブされ、リアルタイムでアクセスできなくなります。

- S3 Glacier Flexible Retrieval ストレージクラス
- S3 Glacier Deep Archive ストレージクラス
- S3 Intelligent-Tiering Archive アクセス階層
- S3 Intelligent-Tiering Deep Archive アクセス階層

アーカイブされたオブジェクトを復元するには、以下の手順を実行する必要があります。

- S3 Glacier Flexible Retrieval ストレージクラスと S3 Glacier Deep Archive ストレージクラスのオブジェクトの場合は、復元リクエストを行い、オブジェクトの一時コピーが使用可能になるまで待機する必要があります。復元されたオブジェクトの一時コピーが作成されると、オブジェクトのストレージクラスは引き続き同じです。(HeadObject または GetObject API オペレーションリクエストは、S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive をストレージクラスとして返します)。
- S3 Intelligent-Tiering アーカイブアクセス階層および Deep Archive アクセス階層にあるオブジェクトの場合、復元リクエストを開始し、オブジェクトが高頻度アクセス階層に移動されるまで待機する必要があります。

Amazon S3 のすべてのストレージクラスの比較の詳細については、[Amazon S3 ストレージクラスを使用する] を参照してください。S3 Intelligent-Tiering の詳細については、「the section called “S3 Intelligent-Tiering の仕組み”」を参照してください。

S3 Glacier からのオブジェクトの復元

S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive を使用した場合、Amazon S3 は指定された期間のみ、オブジェクトの一時コピーを復元します。その期間が終了すると、復元されたオブジェクトのコピーは削除されます。復元されたコピーの有効期限を変更するには、復元リクエストを再発行します。この場合、Amazon S3 は、現在の時刻を基準にして有効期限を更新します。

> **Note**
>
> S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive からアーカイブされたオブジェクトを復元する場合、アーカイブしたオブジェクトと一時的に復元したコピーの両方の費用が発生します。料金については、「Amazon S3 の料金」を参照してください。

S3 Intelligent-Tiering からのオブジェクトの復元

S3 Intelligent-Tiering Archive アクセス階層、または S3 Intelligent-Tiering Deep Archive アクセス階層からオブジェクトの復元を行うと、オブジェクトは S3 Intelligent-Tiering 高頻度アクセス階層に戻ります。連続 30 日が経過した後もオブジェクトにアクセスがなければ、自動的に低頻度アクセス階層に移行します。90 日以上連続でアクセスがない場合、オブジェクトは S3 Intelligent-Tiering Archive アクセス階層に移行します。オブジェクトが 180 日間連続してアクセスされない場合、オブジェクトは Deep Archive アクセス階層に移行します。

> **Note**
>
> S3 Glacier Flexible Retrieval や S3 Glacier Deep Archive のストレージクラスとは異なり、S3 Intelligent-Tiering オブジェクトの復元リクエストは、`Days` 値を受け付けません。

復元リクエストでの S3 バッチ操作の使用

1 つのリクエストで複数の Amazon S3 オブジェクトを復元するには、S3 バッチオペレーションを使用できます。S3 バッチ操作には、オペレーション対象のオブジェクトのリストを指定します。S3 バッチオペレーションは、各 API オペレーションを呼び出して、指定されたオペレーションを実行します。1 つのバッチオペレーションジョブで、エクサバイトのデータを含む数十億ものオブジェクトに対して、指定されたオペレーションを実行できます。

復元時間

Amazon S3 は、復元リクエストで指定された日数をリクエストされた復元が完了した時刻に加算することで、復元されたオブジェクトの有効期限を計算します。結果として得られた時刻は、Amazon S3 によって、深夜の協定世界時 (UTC) の翌日に丸められます。例えば、復元されたオブジェクトのコピーが 2012 年 10 月 15 日午前 10 時 30 分 (UTC) に作成され、復元期間が 3 日間として指定されたとします。この場合、復元されたコピーの有効期限は 2012 年 10 月 19 日 00:00 (UTC) に切れ、その時点で Amazon S3 はオブジェクトコピーを削除します。

復元ジョブが完了するのにかかる時間は、使用するアーカイブストレージクラスまたはストレージ階層、および、迅速 (S3 Glacier Flexible Retrieval と S3 Intelligent-Tiering Archive Access のみで使用可能)、標準、大容量のどの取り出しオプションを指定するかによって変わってきます。詳細については、「アーカイブの取り出しオプション」を参照してください。

Amazon S3 イベント通知を使用して、復元が完了したときに通知を受けることができます。詳細については、「Amazon S3 イベント通知」を参照してください。

トピック

- アーカイブの取り出しオプション
- アーカイブされたオブジェクトの復元

アーカイブの取り出しオプション

以下は、Amazon S3 にアーカイブされたオブジェクトを復元する際に使用できる取り出しオプションです。

- 迅速 – S3 Glacier Flexible Retrieval ストレージクラスまたは S3 Intelligent-Tiering Archive アクセス階層に保存されているデータにすばやくアクセスします。このオプションは、アーカイブのサブセットに対する不定期の緊急リクエストが必要になった場合に利用できます。最大規模のアーカイブオブジェクト (250 MB 以上) を除くすべてのアーカイブオブジェクトについては、迅速取り出しでアクセスしたデータは通常 1〜5 分以内で使用可能になります。

  > **Note**
  >
  > 緊急取得はプレミアム機能であり、緊急リクエストと取得の料金が発生します。Amazon S3 の料金に関する詳細については、「Amazon S3 の料金」を参照してください。

  プロビジョンドキャパシティーを使用すると、S3 Glacier Flexible Retrieval の迅速な取り出しの取得容量を、必要なときに確実に利用できます。詳細については、「プロビジョンドキャパシティー」を参照してください。
- 標準 – 数時間以内に、アーカイブされたどのオブジェクトにもアクセスできます。標準は、取り出しオプションを指定しないで取り出しリクエストを行った場合にデフォルトで適用されます。標準取り出しは、S3 Glacier Flexible Retrieval ストレージクラスまたは S3 Intelligent-Tiering Archive アクセス階層に保存されているオブジェクトで、通常 3〜5 時間以内に終了します。S3 Glacier Deep Archive ストレージクラスまたは S3 Intelligent-Tiering Deep Archive アクセス階層に保存されているオブジェクトの場合、これらの取り出しは通常 12 時間以内に終了します。標準取り出しは、S3 Intelligent-Tiering に保存されたオブジェクトについては無料です。

  > **Note**
  >
  > - S3 Glacier Flexible Retrieval ストレージクラスまたは S3 Intelligent-Tiering Archive アクセス階層に保存されているオブジェクトの場合、S3 バッチオペレーションの復元オペレーションを使用して開始される標準取得は、通常、数分以内に開始され、3 ~ 5 時間以内に終了します。
  > - S3 Glacier Deep Archive ストレージクラスまたは S3 Intelligent-Tiering Deep Archive アクセス階層に保存されているオブジェクトの場合、S3 バッチオペレーションの復元オペ

レーションを使用して開始される標準取得は、通常 9 時間以内に開始され、12 時間以内に終了します。

- 大容量 – Amazon S3 Glacier の最も安価な取り出しオプションを使用してデータにアクセスします。大容量取り出しでは、大量のデータ (ペタバイト規模を含む) を安価に取得できます。

  S3 Glacier Flexible Retrieval ストレージクラスまたは S3 Intelligent-Tiering Archive アクセス階層に保存されているオブジェクトの場合、一括取得は、通常 5 ~ 12 時間以内に終了します。S3 Glacier Deep Archive ストレージクラスまたは S3 Intelligent-Tiering Deep Archive アクセス階層に保存されているオブジェクトの場合、一括取得は、通常 48 時間以内に終了します。

  一括取得は、S3 Glacier Flexible Retrieval または S3 Intelligent-Tiering に保存されたオブジェクトについては無料です。

次の表は、アーカイブの取り出しオプションをまとめたものです。料金については、「Amazon S3 の料金」を参照してください。

`Expedited`、`Standard`、または `Bulk` の取り出しを行うには、RestoreObject REST API リクエストの `Tier` リクエスト要素を、必要なオプションに設定するか、AWS Command Line Interface (AWS CLI) または AWS SDK の同等な値に設定します。プロビジョンドキャパシティーを購入すると、すべての Expedited 取り出しはプロビジョンドキャパシティーを通じて自動的に提供されます。

プロビジョンドキャパシティー

プロビジョンドキャパシティーを使用すると、S3 Glacier Flexible Retrieval の迅速な取り出しの取得容量を、必要なときに確実に利用できます。容量の各単位について 5 分ごとに 3 回以上の迅速取り出しを提供し、1 秒あたり最大 150 メガバイト (MBps) の取り出しスループットを提供します。

ワークロードからデータのサブセットにアクセスする際に非常に高い信頼性と予測可能性が求められる場合は、プロビジョニングされた取り出し容量の購入を検討してください。プロビジョンドキャパシティーがなくても、需要が異常に高い期間は、迅速取り出しを使用できない可能性があります。環境を問わず、どのような場合でも迅速取り出しにアクセスするには、プロビジョンドキャパシティーを購入することをお勧めします。

プロビジョンドキャパシティーユニットは、ユーザーの AWS アカウントに割り当てられます。そのため、バケット所有者ではなく、データの迅速取り出しのリクエスタが、プロビジョンドキャパシティーユニットを購入する必要があります。

プロビジョンドキャパシティーは、Amazon S3 コンソール、Amazon S3 Glacier コンソール、プロビジョンドキャパシティーの購入 REST API オペレーション、AWS SDK、AWS CLI のいずれかを使用して購入できます。プロビジョンドキャパシティーの料金情報については、[Amazon S3 の料金]を参照してください。

S3 Glacier 復元開始リクエストのレート

S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスに保存されているオブジェクトに復元リクエストを開始すると、復元リクエストのクォータが AWS アカウント に適用されます。S3 Glacier は、1 秒あたり最大 1,000 トランザクションのレートで復元リクエストをサポートします。このレートを超えると、有効なリクエストが制限または拒否され、Amazon S3 が `ThrottlingException` エラーを返します。

必要に応じて、S3 バッチオペレーションを使用して、S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスに保存されている多数のオブジェクトを 1 回のリクエストで取得することもできます。詳細については、「Amazon S3 オブジェクトでの大規模なバッチ操作の実行」を参照してください。

アーカイブされたオブジェクトの復元

次のストレージクラスまたは階層の Amazon S3 オブジェクトはアーカイブされ、リアルタイムでアクセスできなくなります。

- S3 Glacier Flexible Retrieval ストレージクラス
- S3 Glacier Deep Archive ストレージクラス
- S3 Intelligent-Tiering Archive アクセス階層
- S3 Intelligent-Tiering Deep Archive アクセス階層

S3 Glacier Flexible Retrieval ストレージクラスまたは S3 Glacier Deep Archive ストレージクラスに保存されている Amazon S3 オブジェクトは、すぐにはアクセスできません。これらのストレージクラスのオブジェクトにアクセスするには、指定された期間 (日数) の間、オブジェクトの一時コピーを S3 バケットに復元する必要があります。オブジェクトの永続的なコピーが必要な場合は、オブジェクトを復元して、Amazon S3 バケット内にそのオブジェクトのコピーを作成します。復元したオブジェクトのコピーは Amazon S3 コンソールではサポートされていません。このタイプのコピー操作には、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用します。コピーを作成してストレージクラスを変更しない限り、オブジェクトは S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスに保存されます。これらのストレージク

ラスの使用方法については、「アクセス頻度の低いオブジェクトのストレージクラス」を参照してください。

S3 Intelligent-Tiering アーカイブアクセスおよび ディープアーカイブアクセス階層のオブジェクトにアクセスするには、復元リクエストを開始しオブジェクトが高頻度アクセス階層に移動するまで待機する必要があります。アーカイブアクセス階層または Deep Archive アクセス階層から復元すると、オブジェクトは高頻度アクセス階層に戻ります。これらのストレージクラスの使用方法については、「アクセスパターンが変化する、またはアクセスパターンが不明なデータを、自動的に最適化するためのストレージクラス」を参照してください。

オブジェクトのアーカイブについては、「アーカイブされたオブジェクトの操作」を参照してください。

> **Note**
>
> - S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive からアーカイブしたオブジェクトを復元する場合、アーカイブしたオブジェクトと一時的に復元したコピーの両方について料金が発生します。
> - S3 Intelligent-Tiering からオブジェクトを復元する場合、標準取得または一括取得での取得料金は発生しません。
> - 既に復元したアーカイブ済みのオブジェクトに対して呼び出されたその後の復元リクエストは、GET リクエストの料金が発生します。料金については、「Amazon S3 の料金」を参照してください。

アーカイブされたオブジェクトの復元

アーカイブされたオブジェクトは、Amazon S3 コンソール、S3 REST API、AWS SDK、AWS Command Line Interface (AWS CLI)、または S3 バッチオペレーションを使用して復元できます。

S3 コンソールの使用

Amazon S3 コンソールを使用してオブジェクトを復元する

次の手順を使用して、S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラス、または S3 Intelligent-Tiering Archive Access または Deep Archive Access ストレージ層にアーカイブされたオブジェクトを復元します。

アーカイブされたオブジェクトを復元するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、復元するオブジェクトが含まれるバケットの名前を選択します。
4. [オブジェクト] リストで復元するオブジェクトを選択し、[アクション] を選択して、[復元の開始] を選択します。
5. S3 Glacier または S3 Glacier Deep Archive から復元する場合は、[復元されたコピーを使用できる日数] ボックスに、アーカイブデータをアクセス可能にする日数を入力します。
6. [保持期間] で次のいずれかの操作を行います。

   - [大容量取り出し] または [標準取り出し] を選択し、[復元の開始] を選択します。
   - [迅速な取り出し] (S3 Glacier Flexible Retrieval または S3 Intelligent-Tiering Archive アクセスのみに使用可能) を選択します。S3 Glacier Flexible Retrieval でオブジェクトを復元する場合、迅速取り出し用にプロビジョンドキャパシティーを購入するかどうかを選択できます。プロビジョンドキャパシティーを購入する場合は、次のステップに進んでください。購入しない場合は、[復元の開始] を選択してください。

   > **Note**
   >
   > S3 Intelligent-Tiering Archive Access と Deep Archive Access 階層からのオブジェクトは、自動的に高頻度アクセス階層に復元されます。

7. (オプション) S3 Glacier Flexible Retrieval でオブジェクトを復元し、迅速取り出しを選択した場合、プロビジョンドキャパシティーを購入するかどうかを選択できます。プロビジョンドキャパシティーは、S3 Glacier Flexible Retrieval のオブジェクトにのみ使用できます。プロビジョンドキャパシティーを持っている場合、[復元の開始] を選択してプロビジョニングされた取り出しを開始します。

   プロビジョンドキャパシティーを持っている場合、すべての迅速取り出しはプロビジョンドキャパシティーにより処理されます。詳細については、「プロビジョンドキャパシティー」を参照してください。

   - プロビジョンドキャパシティーを持っておらず、購入する予定もない場合、[復元の開始] を選択します。

- プロビジョンドキャパシティーはないが、プロビジョンドキャパシティーユニット (PCU) を購入したい場合は、[PCU を購入] を選択してください。[PCU を購入] ダイアログボックスで、購入する PCU の数を選択し、購入を確認して、[PCU を購入] を選択します。[購入完了] メッセージが表示されたら、[復元の開始] を選択してプロビジョニングされた取り出しを開始します。

AWS CLI の使用

S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive からオブジェクトを復元する

次の例では `restore-object` コマンドを使用して、25 日間でバケット *example-s3-bucket* にオブジェクト *dir1/example.obj* を復元します。

```
aws s3api restore-object --bucket example-s3-bucket --key dir1/example.obj --restore-request '{"Days":25,"GlacierJobParameters":{"Tier":"Standard"}}'
```

この例で使用されている JSON 構文が原因で Windows クライアントにエラーが生じる場合は、復元リクエストを以下の構文に置き換えてください。

```
--restore-request Days=25,GlacierJobParameters={"Tier"="Standard"}
```

S3 Intelligent-Tiering Archive Access と Deep Archive Access からオブジェクトを復元する

次の例では `restore-object` コマンドを使用して、バケット *example-s3-bucket* のオブジェクト *dir1/example.obj* を高頻度アクセス階層に復元します。

```
aws s3api restore-object --bucket example-s3-bucket --key dir1/example.obj --restore-request '{}'
```

> **Note**
>
> S3 Glacier Flexible Retrieval や S3 Glacier Deep Archive のストレージクラスとは異なり、S3 Intelligent-Tiering オブジェクトの復元リクエストは、`Days` 値を受け付けません。

復元ステータスを監視する

`restore-object` リクエストのステータスをモニタリングするには、次の `head-object` コマンドを使用します。

```
aws s3api head-object --bucket example-s3-bucket --key dir1/example.obj
```

詳細については、AWS CLI コマンドリファレンスの「restore-object」を参照してください。

REST API の使用

Amazon S3 には、アーカイブされたオブジェクトの復元を開始するための API オペレーションが用意されています。詳細については、「Amazon Simple Storage Service API リファレンス」の「RestoreObject」を参照してください。

AWS SDK の使用

AWS SDK を使って S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive 内にオブジェクトを復元する方法の例については、「AWS SDK または CLI で RestoreObject を使用する」を参照してください。

S3 バッチオペレーションの使用

1 つのリクエストで複数のアーカイブされたオブジェクトを復元するには、S3 バッチオペレーションを使用できます。S3 バッチオペレーションには、オペレーションターゲットのオブジェクトのリストを指定します。S3 バッチオペレーションは、各 API オペレーションを呼び出して、指定されたオペレーションを実行します。1 つのバッチオペレーションジョブで、エクサバイトのデータを含む数十億ものオブジェクトに対して、指定されたオペレーションを実行できます。

バッチオペレーションジョブを作成するには、復元するオブジェクトのみを含むマニフェストが必要です。S3 Inventory を使用してマニフェストを作成することも、必要な情報を含む CSV ファイルを提供することもできます。詳細については、「the section called “マニフェストの指定”」を参照してください。

S3 バッチオペレーションジョブを作成して実行する前に、お客様に代わって S3 バッチオペレーションを実行するためのアクセス権限を Amazon S3 に付与する必要があります。必要なアクセス許可については、「the section called “アクセス許可の付与”」を参照してください。

> **Note**
>
> バッチ操作ジョブは、S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive ストレージクラスオブジェクト、または S3 Intelligent-Tiering Archive Access および Deep Archive Access ストレージ階層オブジェクトのいずれかで操作できます。バッチオペレーションは、同じジョブ内の両方のタイプのアーカイブオブジェクトに対して操作できません。両方のタイプのオブジェクトを復元するには、別個のバッチ操作ジョブを作成する必要があります。

バッチオペレーションを使用してアーカイブオブジェクトを復元する方法の詳細については、「the section called “オブジェクトの復元”」を参照してください。

S3 Initiate Restore Object Batch Operations ジョブを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[バッチ操作] を選択します。
3. [ジョブの作成]を選択します。
4. AWS リージョン については、ジョブを作成する [Region (リージョン)] を選択します。
5. [マニフェストの形式] で、使用するマニフェストのタイプを選択します。
   - [S3 インベントリレポート] を選択した場合Amazon S3 が CSV 形式のインベントリレポートの一部として生成した `manifest.json` オブジェクトへのパスを入力します。最新のバージョンではなく特定のマニフェストバージョンを使用する場合には、オプションでマニフェストオブジェクトのバージョン ID を含めることもできます。
   - [CSV] を選択する場合は、CSV 形式のマニフェストオブジェクトへのパスを入力します。マニフェストオブジェクトは、コンソールで説明される形式に従う必要があります。最新のバージョンではなく特定のバージョンを使用する場合には、オプションでマニフェストオブジェクトのバージョン ID を含めることもできます。
6. [Next] を選択します。
7. [オペレーション] セクションで、[復元] を選択します。
8. [復元] セクションの [ソースを復元] で、[Glacier Flexible Retrieval または Glacier Deep Archive] か [Intelligent-Tiering Archive Access 階層または Deep Archive Access 階層] のいずれかを選択します。

   [Glacier Flexible Retrieval または Glacier Deep Archive] を選択した場合、[復元したコピーが使用できる日数] に数字を入力します。

   [取得階層] の場合、使用する階層を選択します。
9. [Next] を選択します。
10. [追加オプションを設定する] ページで、以下のセクションに情報を入力します。

- [その他のオプション] セクションに、ジョブの説明を入力し、ジョブの優先度番号を指定します。番号が大きいほど、優先度が高いことを表します。詳細については、「the section called “ジョブの優先度の割り当て”」を参照してください。
- [完了レポート] セクションで、バッチオペレーションで完了レポートを作成するかどうかを選択します。完了レポートに関する詳細については、「the section called “完了レポート”」を参照してください。
- [許可] セクションでは、ユーザーに代わってバッチオペレーションを実行するためのアクセス権限を Amazon S3 に付与する必要があります。必要なアクセス許可については、「the section called “アクセス許可の付与”」を参照してください。
- (オプション) [ジョブタグ] セクションで、キーと値のペアにタグを追加します。詳細については、「the section called “タグの使用”」を参照してください。

完了したら、[Next (次へ)] を選択します。

11. [確認] ページで、設定を確認します。変更が必要な場合は、[戻る] を選択します。それ以外の場合は、[ジョブの作成] を選択します。

バッチオペレーションの詳細については、「バッチオペレーションを使ってオブジェクトを復元する」と「S3 バッチオペレーションジョブの作成」を参照してください。

復元ステータスおよび有効期限日の確認

復元リクエストのステータスや有効期限は、Amazon S3 コンソール、Amazon S3 イベント通知、AWS CLI、または Amazon S3 REST API を使用して確認できます。

> **Note**
>
> S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive のストレージクラスから復元されたオブジェクトが保存されるのは、指定した期間のみです。次のプロシージャは、このようなコピーの有効期限を返します。
> S3 Intelligent-Tiering Archive および Deep Archive のアクセス階層から復元されたオブジェクトには有効期限がない代わりに高頻度アクセス階層に戻されます。

S3 コンソールの使用

Amazon S3 コンソールでオブジェクトの復元ステータスと有効期限を確認するには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、復元しているオブジェクトが含まれるバケットの名前を選択します。
4. [オブジェクト]リストで、復元するオブジェクトを選択します。オブジェクトの詳細ページが表示されます。

   - 復元が完了していない場合は、ページの上部に [復元中] セクションが表示されます。
   - 復元が完了した場合は、ページの上部に [復元中] と書かれたセクションが表示されます。S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive から復元する場合、このセクションには [復元の有効期限] も表示されます。Amazon S3 は、この日にアーカイブされたオブジェクトの復元済みコピーを削除します。

Amazon S3 イベント通知の使用

Amazon S3 イベント通知機能を使って、`s3:ObjectRestore:Completed` アクションを使用すると、オブジェクトの復元完了の通知を受け取ることができます。イベント通知を有効にする方法の詳細については、「Amazon SQS、Amazon SNS、および AWS Lambda を使用した通知の有効化」を参照してください。さまざまな `ObjectRestore` イベントタイプの詳細については、「the section called “SQS、SNS、および Lambda でサポートされているイベントタイプ”」を参照してください。

AWS CLI の使用

オブジェクトのリストアステータスと有効期限を AWS CLI コマンドで確認します。

次の例では `head-object` コマンドを使用して、バケット *`example-s3-bucket`* のオブジェクト *`dir1/example.obj`* のメタデータを表示します。復元中のオブジェクトに対してこのコマンドを実行すると、Amazon S3 は復元が進行中であるかどうか、さらには有効期限を返します (該当する場合)。

```
aws s3api head-object --bucket example-s3-bucket --key dir1/example.obj
```

期待される出力 (復元中):

```
{
```

```
    "Restore": "ongoing-request=\"true\"",
    "LastModified": "2020-06-16T21:55:22+00:00",
    "ContentLength": 405,
    "ETag": "\"b662d79adeb7c8d787ea7eafb9ef6207\"",
    "VersionId": "wbYaE2vtOV0iIBXrOqGAJt3fP1cHB8Wi",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {},
    "StorageClass": "GLACIER"
}
```

期待される出力 (復元完了):

```
{
    "Restore": "ongoing-request=\"false\", expiry-date=\"Wed, 12 Aug 2020 00:00:00 GMT
\"",
    "LastModified": "2020-06-16T21:55:22+00:00",
    "ContentLength": 405,
    "ETag": "\"b662d79adeb7c8d787ea7eafb9ef6207\"",
    "VersionId": "wbYaE2vtOV0iIBXrOqGAJt3fP1cHB8Wi",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {},
    "StorageClass": "GLACIER"
}
```

`head-object` の詳細については、「AWS CLI コマンドリファレンス」の「head-object」を参照してください。

REST API の使用

Amazon S3 には、オブジェクトメタデータを取得するための API オペレーションが用意されています。REST API を使用してアーカイブされたオブジェクトの復元ステータスと有効期限を確認するには、「Amazon Simple Storage Service API リファレンス」の「HeadObject」を参照してください。

進行中の復元速度のアップグレード

復元が進行中でも、復元速度をアップグレードできます。

進行中の復元をより高速の階層にアップグレードするには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。

2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、復元するオブジェクトが含まれるバケットの名前を選択します。
4. [オブジェクト]リストで、復元するオブジェクトを選択します。オブジェクトの詳細ページが表示されます。オブジェクトの詳細ページで、[取得階層をアップグレード] を選択します。オブジェクトの復元状況の確認については、[復元ステータスおよび有効期限日の確認] を参照してください。
5. アップグレードする階層を選択し、[復元] を選択します。

S3 オブジェクトロックの使用

S3 オブジェクトロックは、Amazon S3 オブジェクトが一定期間または無期限に削除または上書きされるのを防ぐのに役立ちます。オブジェクトロックは、write-once-read-many (WORM) モデルを使用してオブジェクトを保存します。Object Lock を使用して、WORM ストレージを必要とする規制要件を満たしたり、オブジェクトの変更や削除に対する保護レイヤーを追加したりできます。

> **Note**
>
> S3 オブジェクトロックは、SEC 17a-4、CFTC、および FINRA の各規制に従った環境での使用について、Cohasset Associates によって評価済みです。オブジェクトロックとこのような規制との関連性の詳細については、「Cohasset Associates Compliance Assessment」を参照してください。

オブジェクトロックは、オブジェクトリテンションの管理をするための 2 つの方法、保持期間およびリーガルホールドを提供します。オブジェクトバージョンには、保持期間とリーガルホールド、またはその両方を含めることができます。

- 保持期間 – オブジェクトがロックされたままである一定期間を指定します。個々のオブジェクトに固有の保持期間を設定できます。また、S3 バケットにデフォルトの保持期間を設定できます。バケットポリシーの `s3:object-lock-remaining-retention-days` 条件キーを使用して、許容される最小保持期間と最大保持期間を制限することもできます。これにより、保持期間の範囲を定めて、保持期間がこの範囲より短くなったり長くなったりするのを制限することができます。
- リーガルホールド – リーガルホールドは、保持期間と同じ保護を提供します。ただし、有効期限はありません。代わりに、明示的に削除するまで、リーガルホールドは維持されます。リーガルホールドは、保持期間から独立しており、個々のオブジェクトバージョンに付与されます。

オブジェクトロックは S3 バージョニングが有効になっているバケットでのみ機能します。オブジェクトバージョンをロックすると、Amazon S3 はそのオブジェクトバージョンのメタデータにロック情報を保存します。オブジェクトに保持期間またはリーガルホールドを設定すると、リクエストで指定されたバージョンのみが保護されます。保持期間とリーガルホールドを使用しても、オブジェクトの新しいバージョンの作成が阻止されたり、オブジェクトの上に追加されるマーカーが削除されたりすることはありません。S3 バージョニングの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

同じオブジェクトキー名を持つ既存の保護対象オブジェクトが既に含まれているバケットにオブジェクトを配置すると、Amazon S3 はそのオブジェクトの新しいバージョンを作成します。オブジェクトの既存の保護バージョンは、その保持設定に従ってロックされたままになります。

S3 オブジェクトロックの仕組み

トピック

- 保持期間
- リテンションモード
- リーガルホールド
- S3 Object Lock を使用するためのベストプラクティス
- 必要なアクセス許可

保持期間

保持期間は、一定期間オブジェクトバージョンを保護します。オブジェクトバージョンに保持期間を設定すると、Amazon S3 はオブジェクトバージョンのメタデータにタイムスタンプを保存して、保持期間の有効期限を示します。保持期間が終了すると、オブジェクトバージョンを上書きまたは削除することができます。

個々のオブジェクトバージョンまたはバケットのプロパティに明示的に保存期間を設定して、バケット内のすべてのオブジェクトに自動的に適用されるようにすることができます。オブジェクトバージョンに保持期間を明示的に適用する場合は、オブジェクトバージョンに対して [リテンション期日] を指定します。Amazon S3 はこの日付をオブジェクトバージョンのメタデータにを保存します。

バケットのプロパティに保持期間を設定することもできます。バケットに保持期間を設定する場合、バケット内に配置されたすべてのオブジェクトバージョンを保護する期間を日数または年単位で指定します。バケットにオブジェクトを配置すると、Amazon S3 は、オブジェクトバージョンの作成タ

イムスタンプに指定された期間を追加してオブジェクトバージョンの保持期日を算出します。これにより、オブジェクトバージョンは、オブジェクトバージョンにその保持期間のロックを明示的に配置した場合と同様に保護されます。

> **Note**
>
> 個別の保存モードと期間が明示的に設定されているオブジェクトバージョンをバケット内に PUT すると、オブジェクトバージョンの個々のオブジェクトロック設定が、バケットプロパティの保存設定よりも優先されます。

他のすべての オブジェクトロック設定と同様に、保持期間は個々のオブジェクトバージョンに適用されます。単一のオブジェクトのバージョンが異なれば、リテンションモードや期間も異なる可能性があります。

たとえば、30 日間の保持期間に 15 日のオブジェクトがあり、同じ名前と 60 日間の保持期間を持つオブジェクトを Amazon S3 に PUT するとします。この場合、PUT は正常に実行され、Amazon S3 は保持期間が 60 日の新しいバージョンを作成します。古いバージョンは元の保持期間を維持し、15 日後に削除可能になります。

保持設定をオブジェクトバージョンに適用した後、保持期間を延長することができます。これを行うには、現在設定されているオブジェクトバージョンよりも後の保持期間を持つオブジェクトバージョンの新しいオブジェクトロックリクエストを送信します。Amazon S3 は、既存の保持期間を新しい、より長い期間に置き換えます。オブジェクトの保持期間を設定するアクセス許可を持つユーザーは、オブジェクトバージョンの保持期間を延長できます。保持期間を設定するには、`s3:PutObjectRetention` アクセス許可が必要です。

オブジェクトまたは S3 バケットに保持期間を設定する場合、コンプライアンスまたはガバナンスの 2 つの保持モードのいずれかを選択する必要があります。

リテンションモード

S3 オブジェクトロックでは、オブジェクトに異なるレベルの保護を適用する次の 2 つの保持モードが提供されています。

- コンプライアンスモード
- ガバナンスモード

コンプライアンスモードでは、AWS アカウント の root ユーザーを含め、ユーザーが、保護されたオブジェクトのバージョンを上書きまたは削除することはできません。コンプライアンスモードでオブジェクトをロックすると、そのリテンションモードを変更することはできず、保持期間を短縮することはできません。コンプライアンスモードでは、保持期間中にオブジェクトのバージョンを上書きまたは削除できないようにします。

> **Note**
>
> 保持期限が切れる前にコンプライアンスモードのオブジェクトを削除する唯一の方法は、関連する AWS アカウント を削除することです。

ガバナンスモードでは、特別なアクセス許可を持たない限り、ユーザーはオブジェクトのバージョンの上書きや削除、ロック設定を変更することはできません。ガバナンスモードでは、ほとんどのユーザーがオブジェクトを削除できないように保護できます。ただし必要に応じて、一部のユーザーに保持設定を変更したり、オブジェクトを削除したりするアクセス許可を付与することもできます。ガバナンスモードを使用して、コンプライアンスモードの保持期間を作成する前に、保持期間の設定をテストすることもできます。

ガバナンスモードのリテンション設定を上書きまたは削除するには、ユーザーは `s3:BypassGovernanceRetention` アクセス許可を持っている必要があり、また、ガバナンスモードの上書きを必要とするリクエストで、`x-amz-bypass-governance-retention:true` をリクエストヘッダーとして明示的に含める必要があります。

> **Note**
>
> Amazon S3 コンソールにはデフォルトで、`x-amz-bypass-governance-retention:true` ヘッダーが含まれています。ガバナンスモードで保護されているオブジェクトを削除する場合、`s3:BypassGovernanceRetention` アクセス許可があれば、そのオペレーションは正常に完了します。

リーガルホールド

オブジェクトロックを使用すると、オブジェクトバージョンにリーガルホールドを設定することもできます。保持期間と同様に、リーガルホールドは、オブジェクトバージョンが上書きまたは削除されるのを防ぎます。ただし、リーガルホールドには関連する保持期間はなく、削除するまで有効です。

リーガルホールドは、s3:PutObjectLegalHold アクセス許可を持つ任意のユーザーが自由に適用および解除できます。

リーガルホールドは、保持期間から独立しています。オブジェクトバージョンをリーガルホールドしても、そのオブジェクトバージョンのリテンションモードや保持期間には影響しません。

例えば、オブジェクトバージョンをリーガルホールドして、オブジェクトバージョンも保持期間で保護するとします。保持期間が終了すると、オブジェクトは WORM 保護を失いません。むしろ、リーガルホールドでは、許可されたユーザーが明示的に解除するまでオブジェクトの保護が継続されます。同様に、オブジェクトバージョンに有効な保持期間がある間にリーガルホールドを解除すると、保護期間が終了するまでオブジェクトバージョンは保護されたままになります。

S3 Object Lock を使用するためのベストプラクティス

事前定義された保持期間中にほとんどのユーザーがオブジェクトを削除しないようにするが、同時に特別なアクセス許可を持つ一部のユーザーに保持設定を変更したり、オブジェクトを削除したりできる柔軟性を持たせたい場合は、「ガバナンスモード」の使用を検討してください。

AWS アカウント のルートユーザーを含むユーザーが、事前定義された保持期間中にオブジェクトを削除できないようにしたい場合は、「コンプライアンスモード」の使用を検討してください。このモードは、準拠したデータを保存する必要がある場合に使用できます。

「リーガルホールド」は、オブジェクトを変更不可能な状態に保つ期間が不明な場合に使用できます。これを使用する理由として、データの外部監査が今後予定されていて監査が完了するまでオブジェクトを変更不可能な状態に保ちたい、といったことが考えられます。または、進行中のプロジェクトがあり、プロジェクトが完了するまで変更不可能な状態を保ちたいデータセットを使用している場合もあります。

必要なアクセス許可

オブジェクトロックオペレーションにはアクセス許可が必要です。実行しようとする個別のオペレーションによっては、次の許可のいずれかが必要になる場合があります。

- s3:BypassGovernanceRetention
- s3:GetBucketObjectLockConfiguration
- s3:GetObjectLegalHold
- s3:GetObjectRetention
- s3:PutBucketObjectLockConfiguration

- `s3:PutObjectLegalHold`
- `s3:PutObjectRetention`

Amazon S3 のアクセス許可の説明付きの完全なリストについては、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

アクセス許可で条件を使用する方法の詳細については、「条件キーを使用したバケットポリシーの例」を参照してください。

オブジェクトロックの考慮事項

Amazon S3 オブジェクトロックは、オブジェクトが一定期間または無期限に削除または上書きされるのを防ぐのに役立ちます。

Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して、オブジェクトロック情報を表示したり設定したりできます。S3オブジェクトロック機能全般については、「S3 オブジェクトロックの使用」を参照してください。

> **Important**
>
> - オブジェクトロックを有効にしたバケットを作成した後、オブジェクトロックを無効にしたり、バケットのバージョニングを停止することはできません。
> - S3 オブジェクトロックが有効になっている S3 バケットは、サーバーアクセスログの送信先バケットとしては使用できません。詳細については、「the section called “サーバーアクセスのログ記録”」を参照してください。

トピック

- ロック情報を表示するためのアクセス許可
- ガバナンスモードのバイパス
- S3 レプリケーションでのオブジェクトロックの使用
- AmazonS3 インベントリでのオブジェクトロックの使用
- Object Lock による S3 ライフサイクルポリシーの管理
- Object Lock による削除マーカーの管理
- オブジェクトロックでの S3 Storage Lens の使用

- Object Lock が有効なバケットへのオブジェクトのアップロード
- イベントと通知の設定
- バケットポリシーを使用する保持期間の制限の設定

ロック情報を表示するためのアクセス許可

HeadObject オペレーションまたは GetObject オペレーションを使用して、Amazon S3 オブジェクトバージョンのオブジェクトロックステータスをプログラムで表示できます。いずれのオペレーションでも、指定されたオブジェクトバージョンの保持モード、保持期日、リーガルホールドステータスが返されます。また、S3 インベントリを使用して、S3 バケット内の複数のオブジェクトの Object Lock ステータスを表示できます。

オブジェクトバージョンのリテンションモードおよび保存期間を表示するには、`s3:GetObjectRetention` アクセス許可が必要です。オブジェクトバージョンのリーガルホールドステータスを表示するには、`s3:GetObjectLegalHold` アクセス許可が必要です。バケットのデフォルトの保持期間を表示するには、`s3:GetBucketObjectLockConfiguration` アクセス許可が必要です。S3 オブジェクトロックが有効になっていないバケットに対してオブジェクトロック設定をリクエストしりｔｐ、Amazon S3 はエラーを返します。

ガバナンスモードのバイパス

`s3:BypassGovernanceRetention` アクセス許可がある場合は、ガバナンスモードでロックされているオブジェクトバージョンに対して、保護されていない場合と同様にオペレーションを実行できます。このようなオペレーションには、オブジェクトバージョンの削除、保持期間の短縮以外にも、空のパラメータを指定した新しい `PutObjectRetention` リクエストの発行によるオブジェクトロック保持期間の解除があります。

ガバナンスモードをバイパスするには、このモードをバイパスするリクエストに明示的に示す必要があります。これを行うには、`PutObjectRetention` API オペレーションリクエストに `x-amz-bypass-governance-retention:true` ヘッダーを含めるか、AWS CLI または AWS SDK を介したリクエストで同等のパラメータを使用します。`s3:BypassGovernanceRetention` アクセス許可がある場合、S3 コンソールは、S3 コンソールを通じて行われたリクエストにこのヘッダーを自動的に適用します。

> **Note**
>
> ガバナンスモードをバイパスしても、オブジェクトバージョンのリーガルホールドステータスには影響しません。オブジェクトバージョンでリーガルホールドが有効になっている場

合、リーガルホールドは有効のままで、オブジェクトバージョンの上書きまたは削除のリクエストが避けられます。

S3 レプリケーションでのオブジェクトロックの使用

S3 レプリケーションでオブジェクト ロックを使用すると、S3 バケット間でロックされたオブジェクトとその保持メタデータの自動非同期コピーを有効にすることができます。つまり、レプリケートされたオブジェクトの場合、Amazon S3 はレプリケート元バケットのオブジェクトロック設定を受け取ります。言い換えると、ソースバケットで Object Lock が有効になっている場合は、レプリケート先バケットでも Object Lock が有効になっている必要があります。オブジェクトが (S3 レプリケーションの外部で) レプリケート先バケットに直接アップロードされると、レプリケート先バケットに設定された Object Lock が取得されます。レプリケーションを使用すると、ソースバケット内のオブジェクトは、単一または複数の送信先バケットにレプリケートされます。

オブジェクトロックが有効になっているバケットにレプリケーションを設定するには、S3 コンソール、AWS CLI、Amazon S3 REST API 、または AWS SDK を使用できます。

> **Note**
>
> レプリケーションでオブジェクトロックを使用するには、レプリケーションの設定に使用する AWS Identity and Access Management (IAM) ロールで、ソース S3 バケットに対する 2 つの追加のアクセス許可を付与する必要があります。2 つの新しいアクセス許可とは `s3:GetObjectRetention` と `s3:GetObjectLegalHold` です。ロールに `s3:Get*` アクセス許可がある場合、そのステートメントは要件を満たしています。詳細については、「ライブレプリケーションのアクセス許可の設定」を参照してください。
>
> S3 レプリケーション全般については、「オブジェクトのレプリケーション」を参照してください。
>
> S3 レプリケーションの設定の詳細については、「ライブレプリケーションの設定例」を参照してください。

AmazonS3 インベントリでのオブジェクトロックの使用

定義されたスケジュールに従って S3 バケット内のオブジェクトのリストを作成するように Amazon S3 インベントリを設定できます。オブジェクトの次のオブジェクトロック メタデータを含めるように Amazon S3 インベントリを設定できます。

- 保持期日

- 保持モード
- リーガルホールドステータス

詳細については、「Amazon S3 インベントリ」を参照してください。

Object Lock による S3 ライフサイクルポリシーの管理

オブジェクトライフサイクルの管理設定は、削除マーカーの配置も含めて、保護されたオブジェクトで引き続き正常に機能します。ただし、S3 ライフサイクルの有効期限ポリシーでは、オブジェクトのロックされたバージョンを削除することはできません。Object Lock は、オブジェクトが保存されているストレージクラスに関係なく、またストレージクラス間での S3 ライフサイクル移行全体を通して維持されます。

オブジェクトのライフサイクルの管理の詳細については、「ストレージのライフサイクルの管理」を参照してください。

Object Lock による削除マーカーの管理

保護されたオブジェクトバージョンを削除することはできませんが、そのオブジェクトの削除マーカーを作成することはできます。オブジェクトに削除マーカーを配置しても、オブジェクトや、そのオブジェクトバージョンは削除されません。ただし、オブジェクトが削除されたかのように Amazon S3 をほとんどの方法で動作させます。詳細については、「削除マーカーの使用」を参照してください。

> **Note**
>
> 削除マーカーは、基盤となるオブジェクトでの保持期間またはリーガルホールドに関係なく、WORM 保護されません。

オブジェクトロックでの S3 Storage Lens の使用

オブジェクトロックが有効なストレージバイト数とオブジェクト数のメトリクスを表示するには、Amazon S3 Storage Lens を使用できます。S3 Storage Lens は、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージ分析機能です。

詳細については、「S3 ストレージレンズによるデータ保護」を参照してください。

メトリクスの一覧については、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

Object Lock が有効なバケットへのオブジェクトのアップロード

`Content-MD5` ヘッダーは、Object Lock を使用して設定された保持期間のあるオブジェクトをアップロードするリクエストに必要です。MD5 ダイジェストは、バケットにアップロードした後にオブジェクトの整合性を検証するもう 1 つの方法です。オブジェクトをアップロードした後、Amazon S3 はオブジェクトの MD5 ダイジェストを計算し、指定した値と比較します。リクエストは、2 つのダイジェストが一致した場合にのみ成功します。S3 コンソールは自動的にこのヘッダーを追加しますが、PutObject API を使用するときにこのヘッダーを指定する必要があります。

詳細については、「オブジェクトをアップロードするときに Content-MD5 を使用する」を参照してください。

イベントと通知の設定

Amazon S3 イベント通知を使用して、AWS CloudTrail で、オブジェクトロック設定およびデータへのアクセスと変更を追跡できます。CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」の「AWS CloudTrail とは」を参照してください。

また、Amazon CloudWatch を使用してこのデータに基づいてアラートを生成することもできます。CloudWatch の詳細については、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch とは」を参照してください。

バケットポリシーを使用する保持期間の制限の設定

バケットポリシーを使用してバケットの最小と最大の許容保持期間を設定できます。最大保存期間は 100 年です。

以下の例は、`s3:object-lock-remaining-retention-days` 条件キーを使用して最大保持期間を 10 日間に設定するバケットポリシーを示しています。

```
{
    "Version": "2012-10-17",
    "Id": "SetRetentionLimits",
    "Statement": [
        {
            "Sid": "SetRetentionPeriod",
            "Effect": "Deny",
```

```
            "Principal": "*",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": "arn:aws:s3:::example-s3-bucket1/*",
            "Condition": {
                "NumericGreaterThan": {
                    "s3:object-lock-remaining-retention-days": "10"
                }
            }
        }
    ]
}
```

> **Note**
>
> バケットがレプリケーション設定の送信先バケットである場合は、レプリケーションを使用して作成されるオブジェクトレプリカの最小および最大許容保存期間を設定できます。これを実行するには、バケットポリシーで s3:ReplicateObject アクションを許可する必要があります。レプリケーションのアクセス許可の詳細については、「the section called “アクセス許可のセットアップ”」を参照してください。

バケットポリシーの詳細については、後続のトピックを参照してください。

- 「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。
- オブジェクト操作
- 条件キーを使用したバケットポリシーの例

オブジェクトロックの設定

S3 オブジェクトロックでは、Write Once Read Many (WORM) モデルを使用して Amazon S3 にオブジェクトを保存できます。S3 Object オブジェクトロックを使用して、オブジェクトが固定期間または無期限に削除または上書きされることを防止できます。オブジェクトロック機能全般については、「S3 オブジェクトロックの使用」を参照してください。

オブジェクトをロックする前に、S3 オブジェクトロックを使用するようにバケットを有効にする必要があります。その後、保持期間とリーガルホールド、またはその両方を設定できます。

オブジェクトロックを使用するには、特定のアクセス許可が必要です。さまざまなオブジェクトロックオペレーションに関連するアクセス許可のリストについては、「the section called “必要なアクセス許可”」を参照してください。

> **Important**
>
> - オブジェクトロックを有効にしたバケットを作成した後、オブジェクトロックを無効にしたり、バケットのバージョニングを停止することはできません。
> - S3 オブジェクトロックが有効になっている S3 バケットは、サーバーアクセスログの送信先バケットとしては使用できません。詳細については、「the section called “サーバーアクセスのログ記録”」を参照してください。

トピック

新しい S3 バケット作成時にオブジェクトロックを有効にする

Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して新しい S3 バケットを作成する際にオブジェクトロックを有効にできます。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [Create bucket (バケットの作成)] を選択します。

   [バケットの作成] ページが開きます。
4. [バケット名] にバケットの名前を入力します。

> **Note**
>
> バケット作成後に名前は変更できません。バケットの命名の詳細については、「バケットの名前付け」を参照してください。

5. [リージョン] で、バケットを保存する AWS リージョンを選択します。
6. [オブジェクト所有者] で、アクセスコントロールリスト (ACL) を無効または有効にするかを選択して、バケットにアップロードされたオブジェクトの所有権を制御します。
7. [このバケットのパブリックアクセスブロック設定] で、バケットに適用するブロックパブリックアクセス設定を選択します。
8. [バケットのバージョニング] では、[有効にする] をオンにします。

   オブジェクトロックはバージョニング対応バケットでのみ機能します。
9. (オプション) [Tags] (タグ) では、バケットにタグを追加することを選択できます。タグは、ストレージの分類とコストの割り当てに使用されるキーバリューのペアです。
10. [詳細設定] の下で [オブジェクトロック] を探して、[有効にする] をオンにします。

    オブジェクトロックを有効にすると、このバケット内のオブジェクトは永久にロックされることを認識しておく必要があります。
11. [バケットを作成] を選択します。

AWS CLI の使用

次の `create-bucket` の例では、オブジェクトロックを有効にした *`example-s3-bucket1`* という名前の新しい S3 バケットを作成します。

```
aws s3api create-bucket --bucket example-s3-bucket1 --object-lock-enabled-for-bucket
```

詳細と例については、「AWS CLI コマンドリファレンス」の「create-bucket」を参照してください。

> **Note**
>
> AWS CloudShell を使用してコンソールから AWS CLI コマンドを実行できます。AWS CloudShell はブラウザベースの事前認証されたシェルであり、AWS Management Console

から直接起動できます。詳細については、「AWS CloudShell User Guide」の「CloudShell とは」を参照してください。

REST API の使用

REST API を使用して、オブジェクトロックを有効にした新しい S3 バケットを作成できます。詳細については、「Amazon Simple Storage Service API リファレンス」の「CreateBucket」を参照してください。

AWS SDK の使用

AWS SDK を使用して新しい S3 バケットを作成するときに Object Lock を有効にする方法の例については、「AWS SDK または CLI で `CreateBucket` を使用する」を参照してください。

AWS SDK で現在の Object Lock 設定を取得する方法の例は、「AWS SDK または CLI で `GetObjectLockConfiguration` を使用する」を参照してください。

AWS SDK を使用して別の Object Lock 機能のデモンストレーションを行うインタラクティブシナリオは、「AWS SDK を使用して Amazon S3 オブジェクトロック機能を操作する」を参照してください。

さまざまな AWS SDK の使用に関する一般的な情報については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。

既存の S3 バケットでオブジェクトロックを有効にする

Amazon S3 コンソール、AWS CLI、AWS SDK、または Amazon S3 REST API を使用して、既存の S3 バケットのオブジェクトロックを有効にできます。

S3 コンソールの使用

> **Note**
>
> オブジェクトロックはバージョニング対応バケットでのみ機能します。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、サーバーアクセスのログ記録を有効にするバケットの名前を選択します。
4. [プロパティ] タブを選択します。
5. [プロパティ] の下で、[オブジェクトロック] セクションまで下にスクロールして、[編集] をクリックします。
6. [オブジェクトロック] で [有効にする] をオンにします。

   オブジェクトロックを有効にすると、このバケット内のオブジェクトは永久にロックされることを認識しておく必要があります。
7. [Save changes] (変更の保存) をクリックします。

AWS CLI の使用

次の `put-object-lock-configuration` コマンド例は、*`example-s3-bucket1`* という名前のバケットに 50 日間のオブジェクトロック保持期間を設定します。

```
aws s3api put-object-lock-configuration --bucket example-s3-bucket1 --object-lock-
configuration='{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention":
 { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

詳細と例については、「AWS CLI コマンドリファレンス」の「put-object-lock-configuration」を参照してください。

> **Note**
>
> AWS CloudShell を使用してコンソールから AWS CLI コマンドを実行できます。AWS CloudShell はブラウザベースの事前認証されたシェルであり、AWS Management Console から直接起動できます。詳細については、「AWS CloudShell User Guide」の「CloudShell とは」を参照してください。

REST API の使用

Amazon S3 REST API を使用して、既存の S3 バケットのオブジェクトロックを有効にできます。詳細については、「Amazon Simple Storage Service API リファレンス」の「PutObjectLockConfiguration」を参照してください。

AWS SDK の使用

AWS SDK を使用して既存の S3 バケットに対して Object Lock を有効にする方法の例については、「AWS SDK または CLI で PutObjectLockConfiguration を使用する」を参照してください。

AWS SDK で現在の Object Lock 設定を取得する方法の例は、「AWS SDK または CLI で GetObjectLockConfiguration を使用する」を参照してください。

AWS SDK を使用して別の Object Lock 機能のデモンストレーションを行うインタラクティブシナリオは、「AWS SDK を使用して Amazon S3 オブジェクトロック機能を操作する」を参照してください。

さまざまな AWS SDK の使用に関する一般的な情報については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。

S3 オブジェクトのリーガルホールドを設定または変更する

Amazon S3 コンソール、AWS CLI、AWS SDK、または Amazon S3 REST API を使用して、S3 バケットのオブジェクトロックを設定したり、解除できます。

> **Important**
>
> - オブジェクトにリーガルホールドを設定する場合は、オブジェクトのバケットでオブジェクトロックがすでに有効になっている必要があります。
> - 個別の保存モードと期間が明示的に設定されているオブジェクトバージョンをバケット内に PUT すると、オブジェクトバージョンの個々のオブジェクトロック設定が、バケットプロパティの保存設定よりも優先されます。

詳細については、「the section called “リーガルホールド”」を参照してください。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、リーガルホールドを設定したり変更したりするオブジェクトがあるバケットの名前を選択します。

4. [オブジェクト] リストで、リーガルホールドを設定したり変更したりするオブジェクトを選択します。
5. [オブジェクトプロパティ] ページで、[オブジェクトロックのリーガルホールド] セクションを探して、[編集] をクリックします。
6. [有効にする] をオンにしてリーガルホールドを設定するか、[無効にする] をオンにしてリーガルホールドを解除します。
7. [Save changes] (変更の保存) をクリックします。

AWS CLI の使用

次の `put-object-legal-hold` 例では、*example-s3-bucket1* という名前のバケットのオブジェクト *my-image.fs* にリーガルホールドを設定します。

```
aws s3api put-object-legal-hold --bucket example-s3-bucket1 --key my-image.fs --legal-hold="Status=ON"
```

次の `put-object-legal-hold` 例では、*example-s3-bucket1* という名前のバケットのオブジェクト *my-image.fs* のリーガルホールドを解除します。

```
aws s3api put-object-legal-hold --bucket example-s3-bucket1 --key my-image.fs --legal-hold="Status=OFF"
```

詳細と例については、「AWS CLI コマンドリファレンス」の「put-object-legal-hold」を参照してください。

> **Note**
>
> AWS CloudShell を使用してコンソールから AWS CLI コマンドを実行できます。AWS CloudShell はブラウザベースの事前認証されたシェルであり、AWS Management Console から直接起動できます。詳細については、「AWS CloudShell User Guide」の「CloudShell とは」を参照してください。

REST API の使用

REST API を使用して、オブジェクトのリーガルホールドを設定または変更できます。詳細については、「Amazon Simple Storage Service API リファレンス」の「PutObjectLegalHold」を参照してください。

AWS SDK の使用

AWS SDK を使用してオブジェクトにリーガルホールドを設定する方法の例については、「AWS SDK または CLI で PutObjectLegalHold を使用する」を参照してください。

AWS SDK を使用して現在のリーガスホールドのステータスを取得する方法については、「AWS SDK を使用して Amazon S3 オブジェクトのリーガルホールド設定を取得する」を参照してください。

AWS SDK を使用して別の Object Lock 機能のデモンストレーションを行うインタラクティブシナリオは、「AWS SDK を使用して Amazon S3 オブジェクトロック機能を操作する」を参照してください。

さまざまな AWS SDK の使用に関する一般的な情報については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。

S3 オブジェクトの保持期間を設定または変更する

Amazon S3 コンソール、AWS CLI、AWS SDK、または Amazon S3 REST API を使用して、S3 バケットの保持期間を設定したり変更したりできます。

> **Important**
>
> - オブジェクトに保持期間を設定する場合は、オブジェクトのバケットでオブジェクトロックがすでに有効になっている必要があります。
> - 個別の保存モードと期間が明示的に設定されているオブジェクトバージョンをバケット内に PUT すると、オブジェクトバージョンの個々のオブジェクトロック設定が、バケットプロパティの保存設定よりも優先されます。
> - 保持期限が切れる前にコンプライアンスモードのオブジェクトを削除する唯一の方法は、関連する AWS アカウント を削除することです。

詳細については、「保持期間」を参照してください。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、保持期間を設定したり変更したりするオブジェクトがあるバケットの名前を選択します。
4. [オブジェクト] リストで、保持期間を設定したり変更したりするオブジェクトを選択します。
5. [オブジェクトプロパティ] ページで、[オブジェクトロックの保持] セクションを探して、[編集] をクリックします。
6. [保持] の下で、[有効にする] をオンにして保持期間を設定したり、[無効にする] をオンにして保持期間を解除したりします。
7. [有効にする] をオンにする場合は、[保持モード] で [ガバナンスモード] または [コンプライアンスモード] のいずれかを選択します。詳細については、「リテンションモード」を参照してください。
8. [保持期間] の下で、保持期間を終了する日付を選択します。この間、オブジェクトは WORM で保護され、上書きまたは削除することはできません。詳細については、「保持期間」を参照してください。
9. [Save changes] (変更の保存) をクリックします。

AWS CLI の使用

次の `put-object-retention` 例では、*`example-s3-bucket1`* という名前のバケットのオブジェクト *`my-image.fs`* に保持期間を設定します。

```
aws s3api put-object-retention --bucket example-s3-bucket1 --key my-image.fs --
retention='{ "Mode": "GOVERNANCE", "RetainUntilDate": "2025-01-01T00:00:00" }'
```

詳細と例については、「AWS CLI コマンドリファレンス」の「put-object-retention」を参照してください。

> **Note**
>
> AWS CloudShell を使用してコンソールから AWS CLI コマンドを実行できます。AWS CloudShell はブラウザベースの事前認証されたシェルであり、AWS Management Console から直接起動できます。詳細については、「AWS CloudShell User Guide」の「CloudShell とは」を参照してください。

REST API の使用

REST API を使用して、オブジェクトの保持期間を設定または変更できます。詳細については、「Amazon Simple Storage Service API リファレンス」の「PutObjectRetention」を参照してください。

AWS SDK の使用

AWS SDK を使用してオブジェクトに保持期間を設定する方法の例については、「AWS SDK または CLI で PutObjectRetention を使用する」を参照してください。

AWS SDK を使用してオブジェクトの保持期間を取得する方法の例については、「AWS SDK または CLI で GetObjectRetention を使用する」を参照してください。

AWS SDK を使用して別の Object Lock 機能のデモンストレーションを行うインタラクティブシナリオは、「AWS SDK を使用して Amazon S3 オブジェクトロック機能を操作する」を参照してください。

さまざまな AWS SDK の使用に関する一般的な情報については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。

S3 バケットのデフォルト保持期間を設定または変更する

Amazon S3 コンソール、AWS CLI、AWS SDK、または Amazon S3 REST API を使用して、S3 バケットのデフォルトの保持期間を設定したり変更したりできます。バケットに配置されたすべてのオブジェクトのバージョンを保護する期間を日または年で指定します。

> **Important**
>
> - バケットのデフォルトの保持期間を設定する場合は、バケットでオブジェクトロックがすでに有効になっている必要があります。
> - 個別の保存モードと期間が明示的に設定されているオブジェクトバージョンをバケット内に PUT すると、オブジェクトバージョンの個々のオブジェクトロック設定が、バケットプロパティの保存設定よりも優先されます。
> - 保持期限が切れる前にコンプライアンスモードのオブジェクトを削除する唯一の方法は、関連する AWS アカウント を削除することです。

詳細については、「保持期間」を参照してください。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、デフォルトの保持期間を設定したり変更したりするバケットの名前を選択します。
4. [プロパティ] タブを選択します。
5. [プロパティ] の下で、[オブジェクトロック] セクションまで下にスクロールして、[編集] をクリックします。
6. [保持] の下で、[有効にする] をオンにしてデフォルトの保持期間を設定したり、[無効にする] をオンにしてデフォルトの保持期間を解除したりします。
7. [有効にする] をオンにする場合は、[保持モード] で [ガバナンスモード] または [コンプライアンスモード] のいずれかを選択します。詳細については、「リテンションモード」を参照してください。
8. [デフォルトの保存期間] で、保存期間を有効にする日数または年数を選択します。このバケットに保存されるオブジェクトは、指定した日数または年数の間ロックされます。詳細については、「保持期間」を参照してください。
9. [Save changes] (変更の保存) をクリックします。

AWS CLI の使用

次の put-object-lock-configuration コマンド例は、*example-s3-bucket1* という名前のバケットにコンプライアンスモードを使用して 50 日間のオブジェクトロック保持期間を設定します。

```
aws s3api put-object-lock-configuration --bucket example-s3-bucket1 --object-lock-
configuration='{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention":
 { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

次の put-object-lock-configuration 例では、バケットのデフォルトの保持設定を解除します。

```
aws s3api put-object-lock-configuration --bucket example-s3-bucket1 --object-lock-
configuration='{ "ObjectLockEnabled": "Enabled"}'
```

詳細と例については、「AWS CLI コマンドリファレンス」の「put-object-lock-configuration」を参照してください。

> **Note**
>
> AWS CloudShell を使用してコンソールから AWS CLI コマンドを実行できます。AWS CloudShell はブラウザベースの事前認証されたシェルであり、AWS Management Console から直接起動できます。詳細については、「AWS CloudShell User Guide」の「CloudShell とは」を参照してください。

REST API の使用

REST API を使用して既存の S3 バケットにデフォルトの保持期間を設定できます。詳細については、「Amazon Simple Storage Service API リファレンス」の「PutObjectLockConfiguration」を参照してください。

AWS SDK の使用

AWS SDK を使用して既存の S3 バケットにデフォルトの保持期間を設定する方法の例については、「AWS SDK または CLI で PutObjectLockConfiguration を使用する」を参照してください。

AWS SDK を使用して別の Object Lock 機能のデモンストレーションを行うインタラクティブシナリオは、「AWS SDK を使用して Amazon S3 オブジェクトロック機能を操作する」を参照してください。

さまざまな AWS SDK の使用に関する一般的な情報については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。

Amazon S3 ストレージクラスを使用する

Amazon S3 内にある各オブジェクトには、ストレージクラスが関連付けられています。たとえば、S3 バケット内のオブジェクトをリストすると、コンソールにリスト内のすべてのオブジェクトのストレージクラスが表示されます。Amazon S3 では保存するオブジェクト用の幅広いストレージクラスが提供されています。ユースケースシナリオおよびパフォーマンスアクセス要件を考慮してクラスを選択します。このストレージクラスはすべて高度な耐久性を提供します。

以下のセクションでは、さまざまなストレージクラスの詳細と、オブジェクトにストレージクラスを設定する方法について説明します。

トピック

アクセス頻度の高いオブジェクトのストレージクラス

パフォーマンス重視のユースケース (ミリ秒単位のアクセス時間を要するユースケース) とアクセス頻度の高いデータには、Amazon S3 は次のストレージクラスを提供しています。

- S3 Standard – デフォルトのストレージクラス。オブジェクトのアップロード時にストレージクラスを指定しない場合、Amazon S3 Standard ストレージクラスが割り当てられます。
- S3 Express One Zone – Amazon S3 Express One Zone は、最もレイテンシーの影響を受けやすいアプリケーションに 1 桁のミリ秒単位で一貫したデータアクセスを提供することを目的として構築された、高パフォーマンスのシングルアベイラビリティーゾーンの Amazon S3 ストレージ クラスです。S3 Express One Zone は、現在入手可能なレイテンシーが最も低いクラウドオブジェクトストレージクラスで、データアクセス速度は最大 10 倍速く、リクエストコストは S3 スタンダードよりも 50% 低減されます。S3 Express One Zone では、データは単一のアベイラビリティーゾーン内で冗長的に複数のデバイスに保存されます。詳細については、「S3 Express One Zone とは」を参照してください。
- 低冗長化 – 低冗長化ストレージ (RRS) のストレージクラスは、それほど重要ではない再生可能なデータを、S3 Standard ストレージクラスより低いレベルの冗長性で保存することができます。

> **Important**
>
> このストレージクラスの使用は推奨しません。S3 Standard ストレージクラスの方がコスト効果に優れています。

耐久性について、RRS オブジェクトの平均年間予測喪失率は 0.01 パーセントです。RRS オブジェクトが紛失した場合、Amazon S3 はそのオブジェクトへのリクエストに 405 エラーを返します。

アクセスパターンが変化する、またはアクセスパターンが不明なデータを、自動的に最適化するためのストレージクラス

S3 Intelligent-Tiering は、パフォーマンスへの影響やオペレーション上のオーバーヘッドを発生させることなく、データを最も費用効果の高いアクセス階層に自動的に移動することで、ストレージコストを最適化できるように設計された Amazon S3 ストレージクラスです。S3 Intelligent-Tiering は、アクセスパターンが変更されたときに、アクセス階層間で、細かなオブジェクトレベルでデータを移動することにより、自動的にコスト削減を実現する唯一のクラウドストレージクラスです。S3 Intelligent-Tiering は、アクセスパターンが不明または変化するデータのストレージコストを最小限に抑える場合に最適なストレージクラスです。S3 Intelligent-Tiering には取り出し料金は発生しません。

オブジェクトのモニタリングとオートメーションに対して発生する少額の月額料金で、S3 Intelligent-Tiering はアクセスパターンをモニタリングし、あまりアクセスされていないオブジェクトをより低コストのアクセス階層へ自動的に移動させることができます。S3 Intelligent-Tiering は、3 つの低レイテンシーかつ高スループットのアクセス階層で、自動的にストレージコストを削減できます。非同期でアクセスできるデータの場合は、S3 Intelligent-Tiering ストレージクラスで自動アーカイブ機能をアクティブ化することもできます。S3 Intelligent-Tiering は、99.9% の可用性と 99.999999999% の耐久性を実現するように設計されています。

S3 Intelligent-Tiering は、3 つのアクセス階層にオブジェクトを自動的に保存します。

- 高頻度アクセス – S3 Intelligent-Tiering にアップロードまたは転送されたオブジェクトは、高頻度アクセス階層に自動的に保存されます。
- 低頻度アクセス – S3 Intelligent-Tiering では、30 日間連続してアクセスされなかったオブジェクトは低頻度アクセス階層に移動されます。
- アーカイブインスタントアクセス – S3 Intelligent-Tiering では、90 日間連続してアクセスされていない既存のオブジェクトは、アーカイブインスタントアクセス階層に自動的に移動されます。

S3 Intelligent-Tiering では、これらの 3 つの階層に加えて、オプションとして次の 2 つのアーカイブアクセス階層が用意されています。

- アーカイブアクセス – S3 Intelligent-Tiering では、非同期的にアクセスできるデータ向けにアーカイブアクセス層をアクティブ化するオプションが提供されます。アクティブ化された後、アーカイブアクセス階層は 90 日間以上してアクセスされなかったオブジェクトを自動的にアーカイブします。
- ディープアーカイブアクセス – S3 Intelligent-Tiering では、非同期的にアクセスできるデータ向けにディープアーカイブアクセス階層をアクティブ化するオプションが提供されます。アクティブ化後、ディープアーカイブアクセス階層は 180 日間連続してアクセスされなかったオブジェクトを自動的にアーカイブします。

> **Note**
>
> - アーカイブインスタントアクセス階層をバイパスしたい場合は、アーカイブアクセス階層を 90 日間だけアクティブ化します。アーカイブアクセス階層は、分単位から時間単位の取得時間でストレージコストをわずかに削減します。アーカイブインスタントアクセス階層は、ミリ秒単位のアクセスと高いスループットパフォーマンスを実現します。
> - アプリケーションがオブジェクトに非同期でアクセスできる場合にのみ、アーカイブアクセス階層とディープアーカイブアクセス階層をアクティブにします。取得するオブジェクトがアーカイブアクセス階層またはディープアーカイブアクセス階層に保存されている場合は、まず `RestoreObject` を使用してオブジェクトを復元します。

新しく作成されたデータを S3 Intelligent-Tiering に移動して、デフォルトのストレージクラスとして設定することができます。また、`PutBucketIntelligentTieringConfiguration` API オペレーション、AWS CLI、または Amazon S3 コンソールを使用して、アーカイブアクセス階層の 1 つまたは両方をアクティブ化することもできます。S3 Intelligent-Tiering の使用方法およびアクティブアクセス階層のアクティブ化の詳細については、「S3 Intelligent-Tiering の使用」を参照してください。

Archive Access または Deep Archive Access 階層のオブジェクトにアクセスするには、まずそれらを復元する必要があります。詳細については、「S3 Intelligent-Tiering アーカイブアクセス階層からのオブジェクトの復元」を参照してください。

> **Note**
>
> オブジェクトのサイズが 128 KB 未満の場合は、モニタリングされず、自動階層化に適していません。小さいオブジェクトは必ず高頻度アクセス階層に保存されます。S3 Intelligent-Tiering の詳細については、「S3 Intelligent-Tiering アクセス階層」を参照してください。

アクセス頻度の低いオブジェクトのストレージクラス

S3 Standard – IA と S3 1 ゾーン – IA のストレージクラスは、存続期間が長く、アクセス頻度の低いデータ用に設計されています (IA は infrequent access (低頻度アクセス) の略です)。S3 Standard – IA オブジェクトおよび S3 1 ゾーン – IA オブジェクトは、ミリ秒のアクセスに使用できます (S3 Standard ストレージクラスに似ています)。Amazon S3 では、これらのオブジェクトに取り出し料金が発生するため、アクセスが頻繁ではないデータに最適です。料金については、「Amazon S3 の料金」を参照してください。

例えば、S3 Standard – IA と S3 1 ゾーン – IA のストレージクラスを選択し、次のことを行います。

- バックアップの保存
- アクセスが頻繁ではないが、ミリ秒単位のアクセスを必要とする旧いデータに適しています。例えば、データのアップロード時に S3 Standard ストレージクラスを選択し、Amazon S3 がオブジェクトを S3 Standard – IA または S3 1 ゾーン – IA クラスに移動するようにライフサイクル設定を使用することができます。

  ライフサイクル管理の詳細については、「ストレージのライフサイクルの管理」を参照してください。

> **Note**
>
> S3 Standard – IA と S3 1 ゾーン – IA ストレージクラスは、サイズが 128 KB 以上あり、少なくとも 30 日間保存する予定のオブジェクトに最適です。オブジェクトが 128 KB 以下の場合、Amazon S3 は 128 KB に相当する料金を請求します。最小ストレージ期間の 30 日が過ぎる前にオブジェクトを削除した場合は、30 日分の料金が発生します。30 日前に削除、上書き、または別のストレージクラスに移行されたオブジェクトは、通常のストレージ使用料に加えて、最低 30 日間の残りの期間分の日割り料金が発生します。料金については、「Amazon S3 の料金」を参照してください。

ストレージクラスの違いについては下記をご覧ください。

- S3 Standard-IA – Amazon S3 は、地理的に分離された複数のアベイラビリティーゾーン間でオブジェクトデータを冗長的に保存します (S3 Standard ストレージクラスに似ています)。S3 Standard – IA オブジェクトはアベイラビリティーゾーンに障害が発生した場合の回復性に優れています。このストレージクラスは S3 1 ゾーン – IA クラスよりも優れた高可用性と回復性を提供します。
- S3 One Zone-IA – Amazon S3 はオブジェクトデータを 1 つのアベイラビリティーゾーンでのみ保存するため、S3 Standard-IA よりも安価になります。ただし、地震や洪水といった災害によるアベイラビリティーゾーンの物理的な損失時においてデータの回復性はありません。S3 One Zone-IA ストレージクラスは S3 Standard-IA に並ぶ耐久性を備えていますが、可用性と回復性は劣ります。ストレージクラスの耐久性および可用性の比較情報については、このセクションの最後にある「Amazon S3 ストレージクラスの比較」を参照してください。料金については、「Amazon S3 の料金」を参照してください。

次の構成を推奨します。

- S3 Standard-IA – プライマリまたは再作成できないデータのコピーでのみ使用します。
- S3 One Zone-IA – アベイラビリティーゾーンでの障害発生時にデータを再作成できる場合や、S3 クロスリージョンレプリケーション (CRR) 設定時のオブジェクトレプリカに使用します。

アクセス頻度の低いオブジェクトのストレージクラス

S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval、および S3 Glacier Deep Archive ストレージクラスは、低コストで長期保存するために使用できるデータストレージおよびデータアーカイブ向けに設計されています。これらのストレージクラスは S3 Standard ストレージクラスと S3 標準 – IA クラスと同様の耐久性と回復性を提供します。S3 Glacier ストレージクラスの詳細については、「S3 Glacier ストレージクラスを使用した長期データストレージ」を参照してください。

Amazon S3 は以下の S3 Glacier ストレージクラスを提供します:

- S3 Glacier Instant Retrieval – ほとんどアクセスされず、ミリ秒単位の取得が必要な長期データに使用します。このストレージクラスのデータにはリアルタイムアクセスできます。
- S3 Glacier Flexible Retrieval – データの一部を数分で取得する必要があるアーカイブに使用します。このストレージクラスのデータはアーカイブされ、リアルタイムアクセスできなくなります。
- S3 Glacier Deep Archive – ほとんどアクセスする必要がないデータのアーカイブに使用します。このストレージクラスのデータはアーカイブされ、リアルタイムアクセスできなくなります。

アーカイブ済みオブジェクトの取得

オブジェクトのストレージクラスを S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive に設定するには、セクション オブジェクトのストレージクラスを設定 で説明しているように、他のストレージクラスに対して行う場合と同様の方法で行います。ただし、S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive オブジェクトはアーカイブされ、リアルタイムでアクセスできなくなります。詳細については、「アーカイブストレージ」を参照してください。

> **Note**
>
> S3 Glacier ストレージクラスを使用する場合、オブジェクトは Amazon S3 に残ります。Amazon S3 Glacier サービスを別途使用して直接アクセスすることはできません。Amazon S3 Glacier の詳細については、「Amazon S3 Glacier デベロッパーガイド」を参照してください。

Amazon S3 on Outposts のストレージクラス

Amazon S3 on Outposts を使用すると、AWS Outposts リソースで S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを保存および取得できます。AWS Outposts では、Amazon S3 と同じ API オペレーションと機能 (アクセスポリシー、暗号化、タグ付けを含む) を使用できます。AWS Management Console、AWS CLI、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。

S3 on Outposts は、新しいストレージクラスである S3 Outposts (OUTPOSTS) を提供します。S3 Outposts のストレージクラスは、Outposts のバケットに格納されたオブジェクトに対してのみ使用できます。AWS リージョン の S3 バケットでこのストレージクラスを使用しようとすると、`InvalidStorageClass` エラーが発生します。さらに、S3 on Outposts に保存されているオブジェクトで他の S3 ストレージクラスを使用しようとすると、同じエラー応答が発生します。

S3 Outposts (OUTPOSTS) ストレージクラスに格納されているオブジェクトは、デフォルトでは Amazon S3 マネージド暗号化キー (SSE-S3) によるサーバー側の暗号化を使用して常に暗号化されます。詳細については、「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)」を参照してください。

お客様が用意した暗号化キー (SSE-C) を使用したサーバー側の暗号化を使用して、S3 Outposts ストレージクラスに格納されたオブジェクトを明示的に暗号化することもできます。詳細については、「お客様が指定したキーによるサーバー側の暗号化 (SSE-C) の使用」を参照してください。

> **Note**
>
> S3 on Outposts は、AWS Key Management Service (AWS KMS) キー (SSE-KMS) を使用したサーバー側暗号化をサポートしていません。

S3 on Outposts の詳細については、「Amazon S3 on Outposts とは」を参照してください。

Amazon S3 ストレージクラスの比較

ストレージクラスの可用性、耐久性、最小ストレージ期間、その他の考慮すべき事項を含め、次の表で比較しています。

Storage Class	Designed for	Durability (designed for)	Availability (designed for)	Availability Zones	Min storage duration	Min billable object size	Other Considerations
STANDARD	Frequently accessed data	99.999999999%	99.99%	>= 3	None	None	None
STANDARD_IA	Long-lived, infrequently accessed data	99.999999999%	99.9%	>= 3	30 days	128 KB	Per GB retrieval fees apply.
INTELLIGENT_TIERING	Long-lived data with changing or unknown access patterns	99.999999999%	99.9%	>= 3	30 days	None	Monitoring and automation fees per object apply. No retrieval fees.
ONEZONE_IA	Long-lived, infrequently accessed, non-critical data	99.999999999%	99.5%	1	30 days	128 KB	Per GB retrieval fees apply. Not resilient to the loss of the Availability Zone.
GLACIER	Long-term data archiving with retrieval times ranging from minutes to hours	99.999999999%	99.99% (after you restore objects)	>= 3	90 days	None	Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects.
DEEP_ARCHIVE	Archiving rarely accessed data with a default retrieval time of 12 hours	99.999999999%	99.99% (after you restore objects)	>= 3	180 days	None	Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects.
RRS (Not recommended)	Frequently accessed, non-critical data	99.99%	99.99%	>= 3	None	None	None

* S3 Glacier Flexible Retrieval では、アーカイブされたオブジェクトごとに 40 KB の追加メタデータが必要です。これには、S3 Glacier Flexier Flexible Retrieval レート (データの識別と取得に必要) で請求される 32 KB のメタデータと、S3 スタンダートレートで請求される追加の 8 KB データが含まれます。S3 Glacier Flexible Retrieval にアーカイブされたオブジェクトのユーザー定義名とメタデータを維持するには、S3 スタンダートレートが必要です。ストレージクラスの詳細については、「Amazon S3 ストレージクラス」を参照してください。

** S3 Glacier Deep Archive には、アーカイブされたオブジェクトごとに 40 KB の追加メタデータが必要です。これには、S3 Glacier Deep Archive レート (データの識別と取得に必要) で請求される 32 KB のメタデータと、S3 スタンダートレートで請求される追加の 8 KB データが含まれます。Amazon S3 Glacier Deep Archive にアーカイブされたオブジェクトのユーザー定義の名前とメ

タデータを維持するには、S3 スタンダートレートが必要です。ストレージクラスの詳細については、「Amazon S3 ストレージクラス」を参照してください。

S3 One Zone-IA と S3 Express One Zone を除くすべてのストレージクラスは、災害によるアベイラビリティーゾーンの物理的な損失に対して回復力を持つように設計されていることに注意します。また、アプリケーションシナリオのパフォーマンス要件の他に、コストについても考慮してください。ストレージクラスの料金については「Amazon S3 の料金」をご覧ください。

オブジェクトのストレージクラスを設定

オブジェクトストレージクラスを設定および更新するには、Amazon S3 コンソール、AWS SDK、または AWS Command Line Interface (AWS CLI) を使用できます。これらのアプローチはすべて、Amazon S3 API オペレーションを使用して Amazon S3 にリクエストを送信します。

Amazon S3 API オペレーションは、オブジェクトのストレージクラスを次のように設定 (または更新) することをサポートしています。

- 新しいオブジェクトの作成時にストレージクラスを指定できます。例えば、PUT Object、POST Object、および Initiate Multipart Upload API オペレーションを使用してオブジェクトを作成する場合、`x-amz-storage-class` リクエストヘッダーを追加してストレージクラスを指定します。このヘッダーを追加しない場合は、Amazon S3 では、デフォルトのストレージクラスである S3 Standard を使用します。
- 既に Amazon S3 に保存されているオブジェクトのストレージクラスを別のストレージクラスに変更するには、PUT Object - Copy API オペレーションを使用してオブジェクトのコピーを作成します。ただし、PUT Object - Copy を使用して、S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスに保存されているオブジェクトをコピーすることはできません。S3 1 ゾーン — IA から S3 Glacier Instant Retrieval に移行することもできません。

  同じバケットでオブジェクトをコピーし、同じキー名を使用して次のようにリクエストヘッダーを指定します。

  - `x-amz-metadata-directive` ヘッダーを `COPY` に設定します。
  - `x-amz-storage-class` ヘッダーを、使用するストレージクラスに設定します。

  バージョニングを有効にしたバケットでは、オブジェクトの特定バージョンのストレージクラスを変更することはできません。オブジェクトをコピーすると、Amazon S3 によってオブジェクトに新しいバージョン ID が設定されます。

- オブジェクトのサイズが 160 GB 未満の場合は、Amazon S3 コンソールを使用してオブジェクトのストレージクラスを変更できます。それ以上大きい場合は、S3 Lifecycle の設定を追加すると、オブジェクトのストレージクラスを変更できます。
- Amazon S3 コンソールを使用してユーザー定義タグを持つオブジェクトのストレージクラスを変更する場合は、`s3:GetObjectTagging` のアクセス許可が必要です。ユーザー定義タグがなく、サイズが 16 MB を超えるオブジェクトのストレージクラスを変更する場合は、`s3:GetObjectTagging` アクセス許可も必要です。ターゲットバケットポリシーによって `s3:GetObjectTagging` アクションが拒否されると、オブジェクトのストレージクラスは更新されますが、ユーザー定義タグはオブジェクトから削除され、エラーが発生します。
- バケットに S3 ライフサイクル設定を追加すると、Amazon S3 がオブジェクトのストレージクラスを変更できます。詳細については、「ストレージのライフサイクルの管理」を参照してください。
- レプリケーションを設定する場合、レプリケートされたオブジェクトのストレージクラスを別のストレージクラスに設定できます。ただし、S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスに保存されているオブジェクトをレプリケートすることはできません。詳細については、「レプリケーション設定」を参照してください。

アクセスポリシーのアクセス許可を特定のストレージクラスに制限する

Amazon S3 オペレーションのアクセスポリシーアクセス許可を付与する場合、`s3:x-amz-storage-class` 条件キーを使用して、アップロードされたオブジェクトを保存するときに使用するストレージクラスを制限できます。例えば、`s3:PutObject` アクセス許可を付与すると、オブジェクトのアップロードを特定のストレージクラスに制限できます。ポリシーの例については、「例: オブジェクトのアップロードを特定のストレージクラスのオブジェクトに制限する」を参照してください。

ポリシーでの条件の使用方法および Amazon S3 条件キーの完全なリストについては、以下のトピックを参照してください。

- 「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。
- 条件キーを使用したバケットポリシーの例

S3 Glacier ストレージクラスを使用した長期データストレージ

Amazon S3 には、頻繁にはアクセスされない長期データを保存するための費用対効果の高いソリューションを提供するように設計された複数の S3 Glacier ストレージクラスがあります。S3 Glacier ストレージクラスは、次の 3 種類のストレージクラスに分類されます。

- S3 Glacier Instant Retrieval
- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive

データにアクセスする頻度とデータの取り出し速度に応じて、これらのストレージクラスのいずれかを選択します。各ストレージクラスは S3 Standard ストレージクラスと同様の耐久性と回復性を、より低いストレージコストで提供します。S3 Glacier ストレージクラスの詳細については、https://aws.amazon.com/s3/storage-classes/glacier/ を参照してください。

トピック

S3 Glacier ストレージクラスの比較

各 S3 Glacier ストレージクラスには、すべてのオブジェクトについて、最小ストレージ期間があります。最小ストレージ期間より前にオブジェクトを削除、上書き、または別のストレージクラスに移行した場合、最小ストレージ期間全体の料金が請求されます。

S3 Glacier ストレージクラスの一部はアーカイブ用で、これらのクラスに保存されているオブジェクトはアーカイブされ、リアルタイムアクセスには使用できません。詳細については、「アーカイブストレージ」を参照してください。

取り出し時間が長い低頻度アクセスパターン用に設計されたストレージクラスでは、ストレージコストがより低くなります。料金情報については、https://aws.amazon.com/s3/pricing/ を参照してください。

次の表は、S3 Glacier ストレージクラスを選択する際に検討すべき重要ポイントをまとめたものです。

S3 Glacier Instant Retrieval

S3 Glacier Instant Retrieval は、四半期に一度のアクセス頻度で、ミリ秒単位での取り出しが要求される長期データに使用することをお勧めします。このストレージクラスは、画像ホスティング、ファイル共有アプリケーション、予約時にアクセスする医療記録の保存など、パフォーマンスが重視されるユースケースに最適です。

S3 Glacier Instant Retrieval ストレージクラスは、S3 Standard-IA ストレージクラスと同じレイテンシーとスループットパフォーマンスで、オブジェクトにリアルタイムでアクセスできます。S3 Glacier Instant Retrieval は、 S3 Standard-IA と比較して、ストレージコストは低くなりますが、データアクセスコストは高くなります。

S3 Glacier Instant Retrieval ストレージクラスに保存されるデータの最小オブジェクトサイズは 128 KB です。また、このストレージクラスの最小ストレージ期間は 90 日間です。

S3 Glacier Flexible Retrieval

S3 Glacier Flexible Retrieval は、1 年に 1～2 回のアクセス頻度で、即時アクセスを必要としないアーカイブデータに使用することをお勧めします。S3 Glacier Flexible Retrieval では柔軟な取り出し時間を選択でき、数分から数時間のアクセス時間と、無料の一括取り出しによるコストとのバランスをとるのに役立ちます。このストレージクラスは、バックアップとディザスタリカバリに最適です。

S3 Glacier Flexible Retrieval に保存されたオブジェクトは、アーカイブされ、リアルタイムでアクセスできなくなります。詳細については、「アーカイブストレージ」を参照してください。これらのオブジェクトにアクセスするには、まず復元リクエストを開始し、リクエストの完了時にアクセスできるオブジェクトの一時コピーを作成します。詳細については、アーカイブされたオブジェクトの操作を参照してください。オブジェクトを復元するときは、ユースケースに合わせて取り出し階層を選択でき、復元時間が長くなるほど、コストは低くなります。

S3 Glacier Flexible Retrieval では、次の取り出し階層を使用できます。

- 迅速取り出し — 通常、1～5 分でオブジェクトを復元します。迅速取り出しは、需要に応じて使用します。信頼性が高く、予測可能な復元時間を確保するには、プロビジョンド取り出しキャパシティを購入することをお勧めします。詳細については、「プロビジョンドキャパシティー」を参照してください。

- 標準取り出し — 通常、3～5 時間でオブジェクトを復元します。S3 バッチオペレーションを使用すると、1 分～5 時間以内にオブジェクトを復元します。詳細については、「バッチオペレーションを使ってオブジェクトを復元する」を参照してください。
- 一括取り出し — 通常、5～12 時間以内にオブジェクトを復元します。一括取り出しは無料です。

S3 Glacier Flexible Retrieval ストレージクラスのオブジェクトの最小ストレージ期間は 90 日間です。

S3 Glacier Flexible Retrieval では、オブジェクトごとに 40 KB の追加メタデータが必要です。これには、データの特定と取得に必要な 32 KB のメタデータが含まれ、S3 Glacier Flexible Retrieval のデフォルトレートで課金されます。アーカイブされたオブジェクトのユーザー定義の名前とメタデータを維持するには、追加の 8 KB データが必要で、S3 標準レートで課金されます。

S3 Glacier Deep Archive

アクセス頻度が 1 年に 1 回未満のアーカイブデータには、S3 Glacier Deep Archive を使用することをお勧めします。このストレージクラスは、コンプライアンス要件を満たすためにデータセットを複数年間保持するように設計されており、バックアップやディザスタリカバリ、または取り出しまで最大 72 時間待機できるアクセス頻度の低いデータにも使用できます。S3 Glacier Deep Archive は、AWS で最も低コストのストレージオプションです。

S3 Glacier Deep Archive に保存されたオブジェクトは、アーカイブされ、リアルタイムでアクセスできなくなります。詳細については、「アーカイブストレージ」を参照してください。これらのオブジェクトにアクセスするには、まず復元リクエストを開始し、リクエストの完了時にアクセスできるオブジェクトの一時コピーを作成します。詳細については、アーカイブされたオブジェクトの操作を参照してください。オブジェクトを復元するときは、ユースケースに合わせて取り出し階層を選択でき、復元時間が長くなるほど、コストは低くなります。

S3 Glacier Deep Archive では、次の取り出し階層を使用できます。

- 標準取り出し — 通常、12 時間以内にオブジェクトを復元します。S3 バッチオペレーションを使用すると、9～12 時間以内にオブジェクトを復元します。詳細については、「バッチオペレーションを使ってオブジェクトを復元する」を参照してください。
- 一括取り出し — 通常、標準取り出し階層のコストのごく一部で、48 時間以内にオブジェクトを復元します。

S3 Glacier Deep Archive ストレージクラスのオブジェクトの最小ストレージ期間は 180 日間です。

S3 Glacier Deep Archive には、オブジェクトごとに 40 KB の追加メタデータが必要です。これには、データの特定と取得に必要な 32 KB のメタデータが含まれ、S3 Glacier Deep Archive のデフォルトレートで課金されます。アーカイブされたオブジェクトのユーザー定義の名前とメタデータを維持するには、追加の 8 KB データが必要で、S3 標準レートで課金されます。

アーカイブストレージ

S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive はアーカイブストレージクラスです。つまり、これらのストレージクラスにオブジェクトを保存すると、そのオブジェクトはアーカイブされ、直接アクセスできなくなります。アーカイブされたオブジェクトにアクセスするには、そのオブジェクトの復元リクエストを送信し、サービスによりオブジェクトが復元されるまで待ちます。復元リクエストによりオブジェクトの一時コピーが復元され、このリクエストで指定した期間が終了すると、そのコピーは削除されます。詳細については、「アーカイブされたオブジェクトの操作」を参照してください。

これらのストレージクラスでは、アーカイブされたオブジェクトごとに 40 KB の追加メタデータが必要です。これには、データの特定と取得に必要な 32 KB のメタデータが含まれ、そのストレージクラスのデフォルトレートで課金されます。アーカイブされたオブジェクトのユーザー定義の名前とメタデータを維持するには、追加の 8 KB データが必要で、S3 標準レートで課金されます。

これらのストレージクラスのオブジェクトは、マルチパートアップロードを使用してアップロードすると、S3 Standard ストレージクラス料金で請求されます。詳細については、「マルチパートアップロードと料金」を参照してください。

これらのストレージクラスのアーカイブされたオブジェクトは、AWS リージョンあたりアカウントごとに最大 1,000 トランザクション/秒 (TPS) のオブジェクト復元リクエストで復元できます。

S3 標準ストレージクラスと S3 Glacier サービスの違い

S3 Glacier ストレージクラスは Amazon S3 サービスの一部であり、データをオブジェクトとして S3 バケットに保存します。これらのストレージクラスのオブジェクトは、S3 コンソールを使用するか、S3 API または SDK を使用して、プログラムによって管理できます。S3 Glacier ストレージクラスにオブジェクトを保存すると、高度な暗号化、オブジェクトのタグ付け、S3 ライフサイクル設定などの S3 機能を使用して、データのアクセシビリティとコストを管理できます。

> **Important**
>
> あらゆる長期データには、Amazon S3 サービス内の S3 Glacier ストレージクラスを使用することをお勧めします。

Amazon S3 Glacier (S3 Glacier) サービスは、データをボールト内のアーカイブとして保存する個別のサービスです。このサービスは Amazon S3 機能に対応しておらず、データのアップロードおよびダウンロード操作のコンソールのサポートも提供していません。長期データに S3 Glacier サービスを使用することはお勧めしません。このサービスに保存されたデータは、Amazon S3 サービスからアクセスできません。S3 Glacier サービスの詳細については、「Amazon S3 Glacier デベロッパーガイド」を参照してください。Amazon S3 Glacier サービスから Amazon S3 のストレージクラスにデータを転送するには、AWS ソリューションライブラリの「Amazon S3 Glacier ボールトから Amazon S3 へのデータ転送」を参照してください。

Amazon S3 Intelligent-Tiering

S3 Intelligent-Tiering ストレージクラスは、アクセスパターンが変化した際に、オペレーション上のオーバーヘッドやパフォーマンスへの影響を排除し、データを最もコスト効率の高いアクセス層に自動的に移動させることで、ストレージコストを最適化するように設計されています。オブジェクトのモニタリングとオートメーションに対して発生する少額の月額料金で、S3 Intelligent-Tiering はアクセスパターンをモニタリングし、あまりアクセスされていないオブジェクトをより低コストのアクセス階層へ自動的に移動させることができます。

S3 Intelligent-Tiering は、3 つの低レイテンシーかつ高スループットのアクセス階層で、自動ストレージコスト削減を実現します。非同期でアクセスできるデータの場合は、S3 Intelligent-Tiering ストレージクラスで自動アーカイブ機能をアクティブ化することもできます。S3 Intelligent-Tiering には取り出し料金は発生しません。低頻度のアクセス階層または Archive Instant Access 階層にあるオブジェクトに後からアクセスすると、自動的に高頻度のアクセス階層に戻されます。S3 Intelligent-Tiering ストレージクラスのアクセス階層間でオブジェクトを移動させるときに追加の階層化料金は発生しません。

S3 Intelligent-Tiering は、データレイク、データ分析、新しいアプリケーションなど、オブジェクトのサイズや保持期間に関係なく、アクセスパターンが不明、変更される、または予測不可能なデータに対して推奨されるストレージクラスです。

S3 Intelligent-Tiering の使用方法については、以下のセクションを参照してください。

トピック

- S3 Intelligent-Tiering の仕組み
- S3 Intelligent-Tiering の使用
- S3 Intelligent-Tiering の管理

S3 Intelligent-Tiering の仕組み

Amazon S3 Intelligent-Tiering ストレージクラスは、3 つのアクセス階層に自動的にオブジェクトを保存します。1 つの階層は高頻度アクセス用に最適化され、1 つの低コスト階層は低頻度アクセス用に最適化されます。またもう 1 つの極めてコストの低い階層は、ほとんどアクセスされていないデータに最適化されます。オブジェクトのモニタリングおよびオートメーションにかかる月額料金を抑えるために、S3 Intelligent-Tiering はアクセスパターンをモニタリングし、30 日間連続してアクセスがなかったオブジェクトを、低頻度アクセス階層に自動的に移動します。90 日間アクセスされていない場合に、オブジェクトはパフォーマンスへの影響や運用上のオーバーヘッドなしで、アーカイブインスタントアクセス階層に移動されます。

数分から数時間でアクセスできるデータのストレージコストを最小限に抑えるには、アーカイブ機能を有効にして 2 つのアクセス階層を追加します。オブジェクトをアーカイブアクセス階層、ディープアーカイブアクセス階層、またはその両方に階層を下げることができます。アーカイブアクセスにより、S3 Intelligent-Tiering は 90 日間以上連続してアクセスされていないオブジェクトをアーカイブアクセス階層に移動します。ディープアーカイブアクセスにより、S3 Intelligent-Tiering は 180 日間以上連続してアクセスされていないオブジェクトをディープアーカイブアクセス階層に移動します。どちらの階層でも、必要に応じて非アクセス日数を設定できます。

以下のアクションによるアクセスでは、オブジェクトが下位のアーカイブアクセス階層やディープアーカイブアクセス階層に移動されません。

- Amazon S3 コンソールを通じてオブジェクトをダウンロードまたはコピーする。
- CopyObject または UploadPartCopy を呼び出すか、S3 バッチレプリケーションでオブジェクトをレプリケートする。これらの場合、コピーやレプリケーションオペレーションのソースオブジェクトは上位の階層に移動されます。
- GetObject、PutObject、RestoreObject、CompleteMultipartUpload、ListParts、または SelectObjectContent を呼び出す。

例えば、指定した非アクセス日数 (180 日など) より前に `SelectObjectContent` を通じてオブジェクトがアクセスされると、そのアクションによりタイマーがリセットされます。最後の `SelectObjectContent` リクエストが、指定した日数に達するまでは、オブジェクトはアーカイブアクセス階層またはディープアーカイブアクセス階層に移動しません。

低頻度アクセス階層またはアーカイブインスタントアクセス階層にあるオブジェクトに後からアクセスすると、自動的に高頻度アクセス階層に戻されます。

以下のアクションは、オブジェクトを低頻度アクセス階層やアーカイブインスタントアクセス階層から高頻度アクセス階層に自動的に戻すアクセスを構成します。

- Amazon S3 コンソールを通じてオブジェクトをダウンロードまたはコピーする。
- CopyObject または UploadPartCopy を呼び出すか、バッチレプリケーションでオブジェクトをレプリケートする。これらの場合、コピーやレプリケーションオペレーションのソースオブジェクトは上位の階層に移動されます。
- GetObject、PutObject、RestoreObject、CompleteMultipartUpload、または ListParts を呼び出す。

他のアクションは、オブジェクトを低頻度アクセス階層やアーカイブインスタントアクセス階層から高頻度アクセス階層に自動的に戻すアクセスを構成しません。次に示すのは、そのようなアクションのリストのサンプルであり、決定的なものではありません。

- HeadObject、GetObjectTagging、PutObjectTagging、ListObjects、ListObjectsV2、または ListObjectVersions を呼び出す。
- SelectObjectContent の呼び出しは、オブジェクトを上位の高頻度アクセス階層に移動するアクセスを構成しません。また、高頻度アクセス階層から低頻度アクセス階層へのオブジェクトの移動、さらにアーカイブインスタントアクセス階層への移動を妨げません。

S3 Intelligent-Tiering を新しく作成されるデータのデフォルトのストレージクラスとして設定するには、PutBucketIntelligentTieringConfiguration リクエストヘッダーで `INTELLIGENT-TIERING` を指定します。S3 Intelligent-Tiering は、99.9% の可用性と99.999999999% の耐久性を実現するように設計されています。

> **Note**
>
> オブジェクトのサイズが 128 KB 未満の場合は、モニタリングされず、自動階層化に適していません。小さいオブジェクトは必ず高頻度アクセス階層に保存されます。

S3 Intelligent-Tiering アクセス階層

次のセクションでは、さまざまな自動アクセス階層とオプションのアクセス階層について説明します。オブジェクトがアクセス階層間を移動しても、ストレージクラス (S3 Intelligent-Tiering) は変わりません。

高頻度アクセス階層 (自動)

これは、S3 Intelligent-Tiering で作成されたオブジェクトや S3 Intelligent-Tiering に移行したオブジェクトのライフサイクルを開始するデフォルトのアクセス階層です。オブジェクトは、アクセスされている限り、この階層に残ります。高頻度アクセス階層は、低レイテンシーと高スループットのパフォーマンスを提供します。

低頻度アクセス階層 (自動)

オブジェクトが 30 日間連続してアクセスされない場合、オブジェクトは低頻度アクセス階層に移行します。低頻度アクセス階層は、低レイテンシーと高スループットのパフォーマンスを提供します。

アーカイブインスタントアクセス階層 (自動)

オブジェクトが 90 日間連続してアクセスされない場合、オブジェクトはアーカイブインスタントアクセス階層に移行します。アーカイブインスタントアクセス階層は、低レイテンシーと高スループットのパフォーマンスを提供します。

アーカイブアクセス階層 (オプション)

S3 Intelligent-Tiering では、非同期的にアクセスできるデータのためにアーカイブアクセス層をアクティブ化するオプションが提供されます。アクティブ化された後、アーカイブアクセス階層は 90 日間以上してアクセスされなかったオブジェクトを自動的にアーカイブします。アーカイブの最終アクセス時間は、最大 730 日間まで延長できます。アーカイブアクセス階層は、S3 Glacier Flexible Retrieval ストレージクラスと同じパフォーマンスです。

このアクセス階層の標準取得時間は、3 ~ 5 時間の範囲です。S3 バッチオペレーションを使用して復元リクエストを開始した場合、復元は数分以内に開始されます。取り出しオプションの詳細については、「the section called “S3 Intelligent-Tiering アーカイブアクセス階層からのオブジェクトの復元”」を参照してください。

> **Note**
>
> アーカイブインスタントアクセス階層をバイパスしたい場合は、アーカイブアクセス階層を 90 日間だけアクティブ化します。アーカイブアクセス階層は、数分から数時間の取得時間により、ストレージコストをわずかに削減します。アーカイブインスタントアクセス階層は、ミリ秒単位のアクセスと高いスループットパフォーマンスを実現します。

ディープアーカイブアクセス階層（オプション）

S3 Intelligent-Tiering では、非同期的にアクセスできるデータのために ディープアーカイブアクセス階層をアクティブ化するオプションが提供されます。アクティブ化後、ディープアーカイブアクセス階層は 180 日間連続してアクセスされなかったオブジェクトを自動的にアーカイブします。アーカイブの最終アクセス時間は、最大 730 日間まで延長できます。ディープアーカイブアクセス階層のパフォーマンスは、S3 Glacier Deep Archive ストレージクラスと同じです。

このアクセス層のオブジェクトの標準取得は、12 時間以内に行われます。S3 バッチオペレーションを使用して復元リクエストを開始した場合、復元は 9 時間以内に開始されます。取り出しオプションの詳細については、「the section called “S3 Intelligent-Tiering アーカイブアクセス階層からのオブジェクトの復元”」を参照してください。

> **Note**
>
> アプリケーションがオブジェクトに非同期でアクセスできる場合にのみ、アーカイブアクセス階層とディープアーカイブアクセス階層をアクティブにします。取得するオブジェクトがアーカイブアクセス階層またはディープアーカイブアクセス階層に保存されている場合は、まず `RestoreObject` オペレーションを使用してオブジェクトを復元する必要があります。

S3 Intelligent-Tiering の使用

S3 Intelligent-Tiering ストレージクラスを使用すると、ストレージコストを自動的に最適化できます。S3 Intelligent-Tiering は、アクセスパターンが変更されたときに、アクセス階層間で、細かなオブジェクトレベルでデータを移動することにより、自動的にコスト削減を実現します。非同期でアクセスできるデータの場合は、AWS Management Console、AWS CLI、または Amazon S3 API を使用して、S3 Intelligent-Tiering ストレージクラスで自動アーカイブ機能をアクティブ化することもできます。

S3 Intelligent-Tiering へのデータの移行

S3 Intelligent-Tiering にデータを移動するには、2つの方法があります。`x-amz-storage-class` ヘッダーで `INTELLIGENT_TIERING` を指定して、直接データを PUT することができます。あるいは、S3 Standard または S3 標準頻度から S3 Intelligent-Tiering にオブジェクトを移動させるように S3 ライフサイクル設定を設定します。

Direct Put を使用した S3 Intelligent-Tiering へのデータのアップロード

PUT API オペレーションを使用して S3 Intelligent-Tiering ストレージクラスにオブジェクトをアップロードする場合、x-amz-storage-class リクエストヘッダーで S3 Intelligent-Tiering を指定します。

次のリクエストは、イメージ、my-image.jpg を myBucket バケットに保存します。このリクエストでは、x-amz-storage-class ヘッダーを使用して、オブジェクトが S3 Intelligent-Tiering ストレージクラスを使用して保存されるようにリクエストできます。

Example

```
PUT /my-image.jpg HTTP/1.1
Host: myBucket.s3.<Region>.amazonaws.com (http://amazonaws.com/)
Date: Wed, 1 Sep 2021 17:50:00 GMT
Authorization: authorization string
Content-Type: image/jpeg
Content-Length: 11434
Expect: 100-continue
x-amz-storage-class: INTELLIGENT_TIERING
```

S3 ライフサイクルを使用した S3 標準または S3 Standard-低頻度アクセスから S3 Intelligent-Tiering へのデータの移行

S3 ライフサイクルの設定にルールを追加して、オブジェクトをあるストレージクラスから別のストレージクラスに移行するように Amazon S3 に指示することができます。サポートされている移行および関連する制約の詳細については、「S3 ライフサイクルを使用したオブジェクトの移行」を参照してください。

S3 ライフサイクル設定は、バケットレベルまたはプレフィックスレベルで指定できます。この S3 ライフサイクル設定ルールでは、フィルターはキープレフィックス (documents/) を指定しています。したがって、ルールは、キー名プレフィックスが documents/ のオブジェクト (documents/doc1.txt、documents/doc2.txt など) に適用されます。このルールは、作成から 0 日後にオブジェクトを S3 Intelligent-Tiering ストレージクラスに移行するように Amazon S3 に指示する Transition アクションを指定します。この場合、オブジェクトは、作成後の午前 0 時 (UTC) に S3 Intelligent-Tiering に移行できます。

Example

```
<LifecycleConfiguration>
  <Rule>
```

```
    <ID>ExampleRule</ID>
    <Filter>
       <Prefix>documents/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>INTELLIGENT_TIERING</StorageClass>
    </Transition>
 </Rule>
</LifecycleConfiguration>
```

S3 Intelligent-Tiering のアーカイブアクセスおよびディープアーカイブアクセスの有効化

数分から数時間でアクセス可能なデータのストレージコストを最低限に抑えるには、AWS Management Console、AWS CLI、または Amazon S3 API を使用してバケットレベル、プレフィックスレベル、またはオブジェクトタグレベルの設定を作成することで、アーカイブアクセス階層の 1 つまたは両方を有効にすることができます。

S3 コンソールの使用

S3 Intelligent-Tiering 自動アーカイブを有効にするには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. [Buckets (バケット)] リストで、目的のバケットの名前を選択します。
3. [プロパティ] を選択します。
4. S3 Intelligent-Tiering アーカイブの設定セクションに移動し、[設定を作成] を選択します。
5. アーカイブ構成の設定セクションで、S3 Intelligent-Tiering アーカイブ設定に分かりやすい設定名を指定します。
6. [設定範囲の選択] で、使用する設定範囲を選択します。オプションとして、共有プレフィックス、オブジェクトタグ、またはその組み合わせを使用して、バケット内の指定されたオブジェクトの設定範囲を制限することもできます。

    a. 設定の範囲を制限するには、[1 つ以上のフィルターを使用して、この設定の範囲を制限] を選択します。
    b. 単一のプレフィックスを使用して設定の範囲を制限するには、[プレフィックス] の下にプレフィックスを入力します。

c. オブジェクトタグを使用して設定の範囲を制限するには、[タグの追加] を選択し、キーの値を入力します。

7. [ステータス] で [有効] を選択します。
8. [アーカイブ設定] セクションで、有効にするアーカイブアクセス階層の 1 つまたは両方を選択します。
9. [Create] (作成) を選択します。

AWS CLI を使用する場合

次の AWS CLI コマンドを使用して、S3 Intelligent-Tiering 設定を管理できます。

- delete-bucket-intelligent-tiering-configuration
- get-bucket-intelligent-tiering-configuration
- list-bucket-intelligent-tiering-configurations
- put-bucket-intelligent-tiering-configuration

AWS CLI をセットアップする手順については、[AWS CLI を使用した Amazon S3 での開発] を参照してください。

AWS CLI を使用する場合、設定を XML ファイルとして指定することはできません。代わりに JSON を指定する必要があります。サンプル XML S3 Intelligent-Tiering 設定と、AWS CLI コマンドで指定できる同等の JSON を以下に示します。

次の例では、指定したバケットに S3 Intelligent-Tiering 設定を配置します。

Example **put-bucket-intelligent-tiering-configuration**

JSON

```
{
  "Id": "string",
  "Filter": {
    "Prefix": "string",
    "Tag": {
      "Key": "string",
      "Value": "string"
    },
    "And": {
```

```
      "Prefix": "string",
      "Tags": [
        {
          "Key": "string",
          "Value": "string"
        }
        ...
      ]
    }
  },
  "Status": "Enabled"|"Disabled",
  "Tierings": [
    {
      "Days": integer,
      "AccessTier": "ARCHIVE_ACCESS"|"DEEP_ARCHIVE_ACCESS"
    }
    ...
  ]
}
```

XML

```
PUT /?intelligent-tiering&id=Id HTTP/1.1
Host: Bucket.s3.amazonaws.com
<?xml version="1.0" encoding="UTF-8"?>
<IntelligentTieringConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Id>string</Id>
   <Filter>
      <And>
         <Prefix>string</Prefix>
         <Tag>
            <Key>string</Key>
            <Value>string</Value>
         </Tag>
         ...
      </And>
      <Prefix>string</Prefix>
      <Tag>
         <Key>string</Key>
         <Value>string</Value>
      </Tag>
   </Filter>
   <Status>string</Status>
```

```
   <Tiering>
      <AccessTier>string</AccessTier>
      <Days>integer</Days>
   </Tiering>
   ...
</IntelligentTieringConfiguration>
```

PUT API オペレーションの使用

指定したバケットとバケットごとに最大 1,000 の S3 Intelligent-Tiering 設定に対して[PutBucketIntelligentTieringConfiguration](#)オペレーションを使用できます。共有プレフィックスまたはオブジェクトタグを使用して、アーカイブアクセス階層の対象となるバケット内のオブジェクトを定義できます。共有プレフィックスまたはオブジェクトタグを使用して、特定のビジネスアプリケーション、ワークフロー、または内部組織に合わせることができます。また、アーカイブアクセス階層、ディープアーカイブアクセス階層、またはその両方を柔軟にアクティブにすることもできます。

S3 Intelligent-Tiering を使用した開始方法

S3 Intelligent-Tiering の使用方法の詳細については、「[チュートリアル: S3 Intelligent-Tiering の使用を開始する](#)」を参照してください。

S3 Intelligent-Tiering の管理

S3 Intelligent-Tiering ストレージクラスは、3 つの低レイテンシーかつ高スループットのアクセス階層で、ストレージコストの自動的な削減を実現します。また、数分から数時間でアクセス可能なデータに対して最小限のストレージコストを実現できるように、オプションのアーカイブ機能が提供されます。S3 Intelligent-Tiering ストレージクラスは、以下を含む Amazon S3 のすべての機能をサポートします。

- オブジェクトのアクセス層を検証する S3 インベントリ
- データを任意の AWS リージョン にレプリケートする S3 レプリケーション
- S3 Storage Lens でのストレージの使用状況とアクティビティに関するメトリクスの表示
- オブジェクトデータ保護のためのサーバー側の暗号化
- データを誤って削除しないようにする S3 オブジェクトロック
- 仮想プライベートクラウド (VPC) のプライベートエンドポイントを介して Amazon S3 にアクセスするための AWS PrivateLink

どの S3 Intelligent-Tiering アクセス階層オブジェクトが保存されているかを特定する

オブジェクトとオブジェクトに対応するメタデータ (S3 Intelligent-Tiering アクセス層を含む) のリストを取得するには、the section called “インベントリの管理” を使用できます。S3 インベントリは、CSV、ORC、またはオブジェクトとそれに対応するメタデータを一覧表示する Parquet 出力ファイルを提供します。Amazon S3 バケットまたは共有プレフィックスについて、これらのインベントリレポートを日単位または週単位で受け取ることができます。(共有プレフィックスとは、共通の文字列で始まる名前を持つオブジェクトのことです。)

S3 Intelligent-Tiering のオブジェクトのアーカイブステータスを見る

S3 Intelligent-Tiering ストレージクラス内のオブジェクトが、Archive Access tier または Deep Archive Access 階層のいずれかに移動したときに通知を受け取るには、Amazon S3 イベント通知を設定できます。詳細については、イベント通知を有効にするを参照してください。

Amazon S3 は、Amazon Simple Notification Service (Amazon SNS) トピック、Amazon Simple Queue Service (Amazon SQS) キュー、または AWS Lambda 関数にイベント通知を発行できます。詳細については、「Amazon S3 イベント通知」を参照してください。

次は Amazon S3 が `s3:IntelligentTiering` イベントを発行するために送信するメッセージの例です。詳細については、「the section called “イベントメッセージの構造”」を参照してください。

```
{
   "Records":[
      {
         "eventVersion":"2.3",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"IntelligentTiering",
         "userIdentity":{
            "principalId":"s3.amazonaws.com"
         },
         "requestParameters":{
            "sourceIPAddress":"s3.amazonaws.com"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
```

```
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"mybucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::mybucket"
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
            }
         },
         "intelligentTieringEventData":{
            "destinationAccessTier": "ARCHIVE_ACCESS"
            }
      }
   ]
}
```

また、HEAD オブジェクトリクエストを使用して、オブジェクトのアーカイブのステータスを表示することもできます。オブジェクトが S3 Intelligent-Tiering ストレージクラスを使用して保存され、現在アーカイブ層の 1 つにある場合、HEAD オブジェクト応答には現在のアーカイブ層が表示されます。アーカイブ階層を表示するために、リクエストでは x-amz-archive-statusヘッダーを使用します。

次の HEAD オブジェクトリクエストは、オブジェクトのメタデータ (この場合は *my-image.jpg*) を返します。

Example

```
HEAD /my-image.jpg HTTP/1.1
Host: bucket.s3.region.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:02236Q3V0RonhpaBX5sCYVf1bNRuU=
```

また、HEAD オブジェクトリクエストは、restore-object リクエストのステータスをモニタリングするためにも使用できます。アーカイブの復元が進行中の場合、HEAD オブジェクトの応答には x-amz-restore ヘッダーが含まれます。

以下は、復元リクエストの処理中に S3 Intelligent-Tiering を使用してアーカイブされたオブジェクトを示す HEAD オブジェクト応答のサンプルです。

Example

```
HTTP/1.1 200 OK
x-amz-id-2: FSVaTMjrmBp3Izs1NnwBZeu7M19iI8UbxMbi0A8AirHANJBo+hEftBuiESACOMJp
x-amz-request-id: E5CEFCB143EB505A
Date: Fri, 13 Nov 2020 00:28:38 GMT
Last-Modified: Mon, 15 Oct 2012 21:58:07 GMT
ETag: "1accb31fcf202eba0c0f41fa2f09b4d7"
x-amz-storage-class: 'INTELLIGENT_TIERING'
x-amz-archive-status: 'ARCHIVE_ACCESS'
x-amz-restore: 'ongoing-request="true"'
x-amz-restore-request-date: 'Fri, 13 Nov 2020 00:20:00 GMT'
Accept-Ranges: bytes
Content-Type: binary/octet-stream
Content-Length: 300
Server: AmazonS3
```

S3 Intelligent-Tiering アーカイブアクセス階層からのオブジェクトの復元

S3 Intelligent-Tiering Archive Access および Deep Archive のアクセス階層のオブジェクトにアクセスするには、復元リクエストを開始して、オブジェクトが高頻度アクセス階層に移動するまで待機する必要があります。アーカイブされたオブジェクトの詳細については、「the section called “アーカイブされたオブジェクトの操作”」を参照してください。

アーカイブアクセス階層または Deep Archive アクセス階層から復元すると、オブジェクトは高頻度アクセス階層に戻ります。その後、連続 30 日が経過した後もオブジェクトにアクセスがなければ、自動的に低頻度アクセス階層に移行します。次に、90 日以上連続でアクセスされないと、オブジェクトが Archive Access 階層に移行します。180 日以上連続でアクセスされないと、オブジェクトは Deep Archive Access 階層に自動的に移行します。詳細については、「the section called “S3 Intelligent-Tiering の仕組み”」を参照してください。

アーカイブされたオブジェクトは、Amazon S3 コンソール、S3 バッチオペレーション、Amazon S3 REST API、AWS SDK、または AWS Command Line Interface (AWS CLI) を使用して復元できます。詳細については、「the section called “アーカイブされたオブジェクトの操作”」を参照してください。

ストレージのライフサイクルの管理

オブジェクトがライフサイクル全体にわたってコスト効率に優れた方法で保存されるように管理するには、Amazon S3 ライフサイクル設定を作成します。Amazon S3 ライフサイクル設定とは、Amazon S3 がオブジェクトのグループに適用するアクションを定義するルールセットです。次の 2 種類のアクションがあります。

- Transition actions — 別のストレージクラスにオブジェクトを移行するタイミングを定義します。例えば、作成から 30 日後に S3 標準 – IA ストレージクラスにオブジェクトを移行するか、作成から 1 年後に S3 Glacier Flexible Retrieval ストレージクラスにオブジェクトをアーカイブするよう選択することができます。詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。

  ライフサイクル移行リクエストにはコストが発生します。料金については、[Amazon S3 の料金]を参照してください。
- [有効期限切れアクション] — このアクションではオブジェクトの有効期限を定義します。Amazon S3 はユーザーに代わって有効期限切れのオブジェクトを削除します。

  ライフサイクル有効期限切れコストは、オブジェクトの有効期限が切れるタイミングに応じて異なります。詳細については、「オブジェクトの有効期限」を参照してください。

> **Important**
>
> バケットポリシーを使用して、S3 ライフサイクルルールによる削除や移行を防ぐことはできません。例えば、バケットポリシーがすべてのプリンシパルのすべてのアクションを拒否する場合でも、S3 ライフサイクル設定は通常どおり機能します。

既存のオブジェクトと新しいオブジェクト

ライフサイクル設定をバケットに追加すると、設定ルールは既存のオブジェクトとそれ以降に追加されるオブジェクトの両方に適用されます。例えば、オブジェクトが作成から 30 日後に有効期限を迎えるようにする失効アクションを備えたライフサイクル設定ルールを本日追加すると、Amazon S3 により作成から 30 日以上が経過している既存のオブジェクトがすべて削除キューに追加されます。

請求の変更

オブジェクトがライフサイクルアクションの対象になるときと、Amazon S3 がオブジェクトを転送または期限切れにするまでの間に遅延がある場合は、オブジェクトがライフサイクルアクションの対象となり次第すぐに請求の変更が適用されます。例えば、オブジェクトの有効期限がスケジュール済みであり、Amazon S3 がそのオブジェクトを直ちに有効期限切れにしない場合、有効期限が過ぎてもストレージに対する料金は発生しません。

この動作の例外として、S3 Intelligent-Tiering ストレージクラスに移行するライフサイクルルールがある場合があります。オブジェクトが S3 Intelligent-Tiering に移行するまで、請求の変更は行われません。

S3 ライフサイクルルールの詳細については、[ライフサイクル設定の要素] を参照してください。

ライフサイクルルールの影響のモニタリング

アクティブなライフサイクルルールによって行われた更新の影響をモニタリングするには、「the section called “ライフサイクルルールによって実行されたアクションをモニタリングするにはどうすればよいですか？”」を参照してください。

オブジェクトのライフサイクルの管理

ライフサイクルが明確に定義されているオブジェクトの S3 ライフサイクル設定ルールを定義します。例:

- 定期的なログをバケットにアップロードする場合、アプリケーションは 1 週間または 1 か月間、それを必要とする可能性があります。その後は、削除することができます。
- ドキュメントには、一定の期間中に頻繁にアクセスされるものがあります。その後は、頻繁にアクセスされません。ある時点で、リアルタイムでアクセスする必要はないものの、所属している組織や規則によって、特定の期間アーカイブしておくよう要求される場合があります。その期間が過ぎれば、削除してかまいません。
- 主にアーカイブ目的のため、ある種類のデータを Amazon S3 にアップロードする場合があります。たとえば、デジタルメディア、財務や医療の記録、生のゲノムシーケンスデータ、データベースの長期バックアップ、法規制準拠のために保管が必要なデータをアーカイブできます。

S3 ライフサイクル設定ルールを使用すると、より安価なストレージクラスへのオブジェクトの移行、またはアーカイブや削除を Amazon S3 に指定できます。

ライフサイクル設定の作成

S3 ライフサイクル設定は、Amazon S3 がオブジェクトの有効期限中にオブジェクトに実行する定義済みのアクションを含む、一連のルールで構成された XML ファイルです。

ライフサイクル設定は、Amazon S3 コンソール、REST API、AWS SDK、AWS Command Line Interface (AWS CLI) を使用して作成できます。詳細については、「バケットにライフサイクル設定を設定する」を参照してください。

Amazon S3 は、バケットでライフサイクル設定を管理するための一連の REST API オペレーションを提供します。Amazon S3 では、ライフサイクルの設定はバケットにアタッチされたライフサイクルサブリソースとして保存されます。詳細については、以下を参照してください。

- PutBucketLifecycleConfiguration
- GetBucketLifecycleConfiguration
- DeleteBucketLifecycle

ライフサイクル設定の作成に関する詳細は、次のトピックを参照してください。

トピック

- Amazon S3 ライフサイクルを使用したオブジェクトの移行
- オブジェクトの有効期限
- バケットにライフサイクル設定を設定する
- ライフサイクルとその他のバケット設定
- ライフサイクルイベント通知の設定
- ライフサイクル設定の要素
- S3 ライフサイクル設定の例

Amazon S3 ライフサイクルを使用したオブジェクトの移行

S3 ライフサイクルの設定にルールを追加して、別の Amazon S3 ストレージクラスにオブジェクトを移行するように Amazon S3 に指定できます。ストレージクラスの詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。この方法で S3 ライフサイクル設定を使用する場合の例を次に示します。

- これらのオブジェクトのアクセス頻度が低いことがわかった場合は、オブジェクトを S3 Standard – IA ストレージクラスに移行できます。
- リアルタイムでアクセスする必要のないオブジェクトは、S3 Glacier Flexible Retrieval ストレージクラスまたは S3 Glacier Deep Archive ストレージクラスへのアーカイブを推奨します。

既存のオブジェクトと新しいオブジェクト

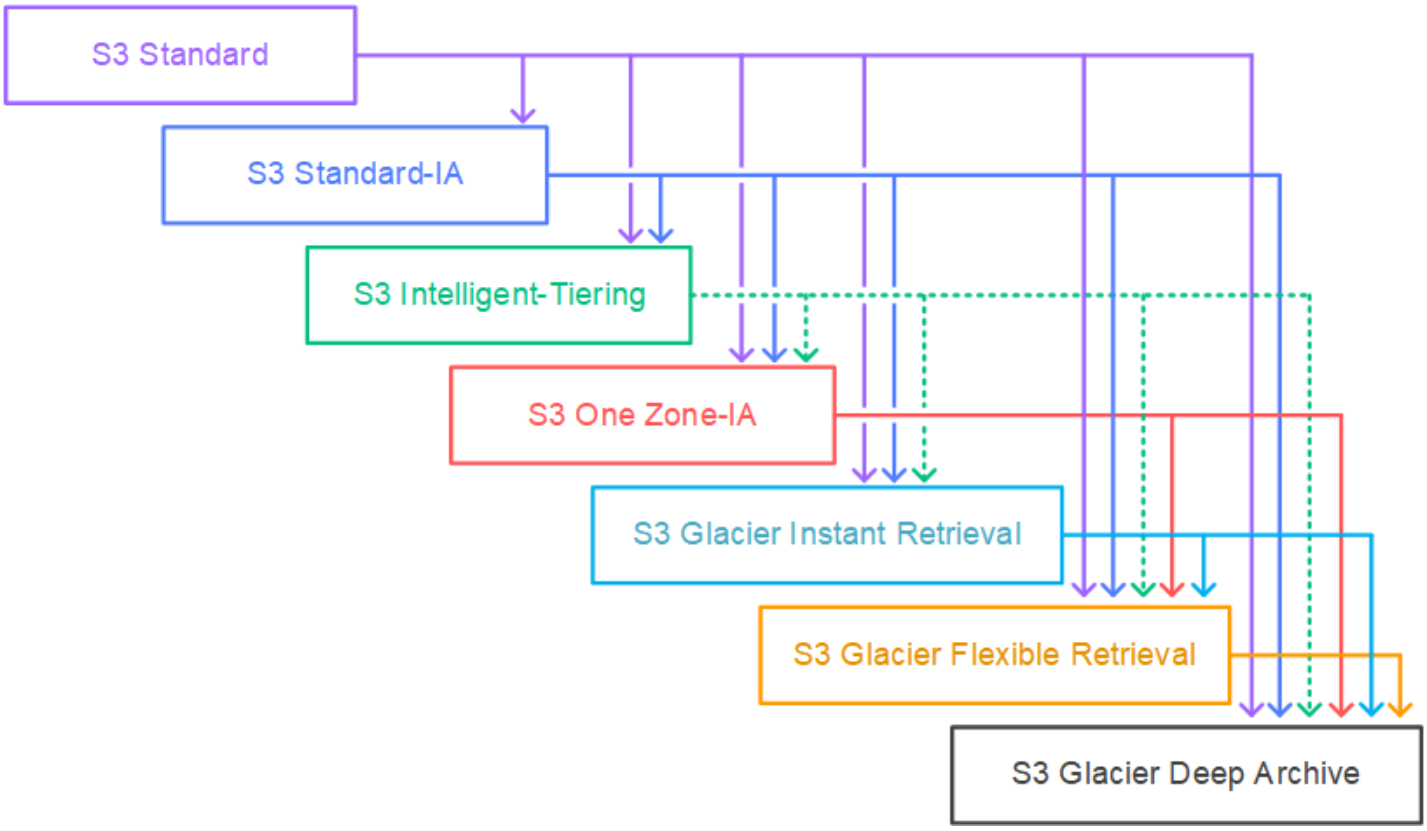
ライフサイクル設定をバケットに追加すると、設定ルールは既存のオブジェクトとそれ以降に追加されるオブジェクトの両方に適用されます。例えば、特定のプレフィックスが付いたオブジェクトが作成後 30 日で別のストレージクラスに移行するようにする移行アクションを備えたライフサイクル設定ルールを本日追加すると、Amazon S3 により作成から 30 日以上が経過し、指定されたプレフィックスを持つ既存のオブジェクトがすべて削除キューに移行されます。

> **Important**
>
> バケットポリシーを使用して、S3 ライフサイクルルールによる削除や移行を防ぐことはできません。例えば、バケットポリシーがすべてのプリンシパルのすべてのアクションを拒否する場合でも、S3 ライフサイクル設定は通常どおり機能します。

サポートされている移行と関連する制約

S3 ライフサイクル設定では、1 つのストレージクラスから別のストレージクラスにオブジェクトを移行し、ストレージのコストを節約できるルールを定義できます。オブジェクトのアクセスパターンが不明、またはアクセスパターンが時間の経過とともに変化している場合、コストを自動的に削減するためにオブジェクトを S3 Intelligent-Tiering ストレージクラスに移行できます。ストレージクラスについては、[Amazon S3 ストレージクラスを使用する] を参照してください。

Amazon S3 は、以下の図のようにストレージクラス間の移行のためのウォーターフォールモデルをサポートします。



サポートされているライフサイクル移行

Amazon S3 では、S3 ライフサイクル設定を使用したストレージクラス間の以下のライフサイクル移行をサポートします。

以下の移行が できます。

- S3 Standard ストレージクラスを他のストレージクラスに移行する。
- S3 標準 – IA ストレージクラスを S3 Intelligent-Tiering、S3 1 ゾーン – IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval、または S3 Glacier Deep Archive ストレージクラスに移行する。
- S3 Intelligent-Tiering ストレージクラスを S3 1 ゾーン – IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval、または S3 Glacier Deep Archive ストレージクラスに移行する。

> **Note**
>
> S3 Intelligent-Tiering ストレージクラスから S3 1 ゾーン – IA ストレージクラスおよび一部の S3 Glacier ストレージクラスにオブジェクトを移行するには、いくつかの例外がありま

す。詳細については、「the section called “サポートされていないライフサイクル移行”」を参照してください。

- S3 One Zone-IA ストレージクラスを S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスに移行する。
- S3 Glacier Instant Retrieval ストレージクラスを S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスに移行する。
- S3 Glacier Flexible Retrieval から S3 Glacier Deep Archive ストレージクラスに移行する。
- 任意のストレージクラスを S3 Glacier Deep Archive ストレージクラスに移行する。

> **Note**
>
> ライフサイクル移行にはデータ取り出し料金はかかりません。ただし、PUT、COPY、またはライフサイクルルールを使用してデータを任意の S3 ストレージクラスに移動する場合、リクエストごとに取り込み料金が発生します。オブジェクトをいずれかのストレージクラスに移動する前に、取り込みコストや移行コストについて検討してください。コストに関する考慮事項の詳細については、「Amazon S3 の料金」を参照してください。

サポートされていないライフサイクル移行

Amazon S3 では、以下のライフサイクル移行はサポートしていません。

以下の移行は できません。

- 任意のストレージクラスを S3 Standard ストレージクラスに移行する。
- 任意のストレージクラスから Reduced Redundancy ストレージクラス (RRS) への移行。
- S3 1 ゾーン – IA ストレージクラスを S3 Intelligent-Tiering、S3 標準 – IA、または S3 Glacier Instant Retrieval ストレージクラスに移行する。
- S3 Intelligent-Tiering ストレージクラス (すべての階層) から S3 Standard–IA ストレージクラス。
- S3 Intelligent-Tiering ストレージクラスのアーカイブインスタントアクセス階層から S3 1 ゾーン-IA。
- S3 Intelligent-Tiering ストレージクラスのアーカイブインスタントアクセス階層から S3 1 ゾーン-IA または S3 Glacier Instant Retrieval。
- S3 Intelligent-Tiering ストレージクラスの Deep Archive アクセス階層から S3 1 ゾーン-IA、S3 Glacier Instant Retrieval、または S3 Glacier Flexible Retrieval。

制約

ライフサイクルストレージクラス移行には、次の制約があります。

オブジェクトサイズと、S3 Standardまたは S3 Standard – IA から S3 Intelligent-Tiering、S3 Standard – IA、または S3 1 ゾーン – IA への移行

S3 Standard または S3 Standard – IA ストレージクラスから S3 Intelligent-Tiering、S3 Standard – IA、または S3 1 ゾーン – IA にオブジェクトを移行する場合、次のオブジェクトサイズ制約が適用されます。

- [大きいオブジェクト] - 今後の移行で、大きなオブジェクトを移行する場合にコスト面での利点があります。
  - S3 Standard または S3 Standard-IA ストレージクラスから S3 Intelligent-Tiering への移行。
  - S3 Standard ストレージクラスから S3 Standard – IA または S3 1 ゾーン – IA。
- 128 KiB 未満のオブジェクト – 以下の移行では、Amazon S3 は 128 KiB 未満のオブジェクトを移行しません。
  - S3 Standard または S3 Standard-IA ストレージクラスから S3 Intelligent-Tiering または S3 Glacier Instant Retrieval への移行。
  - S3 Standard ストレージクラスから S3 Standard – IA または S3 1 ゾーン – IA。

> **Note**
>
> オブジェクトサイズに基づいてライフサイクルルールをフィルタリングできます。

> **Important**
>
> S3 ライフサイクル設定に複数のルールがある場合、1 つのオブジェクトが同じ日に複数の S3 ライフサイクルアクションの対象になることがあります。このような場合、Amazon S3 は以下の一般的なルールに従います。
>
> - 完全な削除は、移行より優先されます。
> - 移行は、削除マーカーの作成より優先されます。
> - オブジェクトが S3 Glacier Flexible Retrieval 移行と S3 標準 – IA 移行 (または S3 1 ゾーン – IA 移行) の両方の対象になる場合、Amazon S3 は S3 Glacier Flexible Retrieval の移行を選択します。

例については、「例 5: 重複するフィルター、競合するライフサイクルアクション、Amazon S3 がバージョン管理されていないバケットに対して行うこと」を参照してください。

S3 標準 – IA 移行または S3 1 ゾーン – IA 移行の最小日数

S3 標準 - IA または S3 1 ゾーン - IA にオブジェクトを移行する前に、これらのオブジェクトを少なくとも 30 日間 Amazon S3 に保存する必要があります。例えば、作成から 1 日後にオブジェクトを S3 標準 – IA ストレージクラスに移行するライフサイクルルールを作成することはできません。Amazon S3 は最初の 30 日間はこの移行をサポートしません。新しいオブジェクトはアクセス頻度が高く、S3 標準 – IA または S3 1 ゾーン – IA ストレージに適した期間よりも早く削除されることが多いためです。

同様に、(バージョニング対応のバケットで) 以前のオブジェクトを移行する場合、少なくとも 30 日は最新でないオブジェクトのみを S3 Standard – IA または S3 1 ゾーン – IA ストレージに移行できます。すべてのストレージクラスの最小ストレージ期間のリストについては、「Amazon S3 ストレージクラスの比較」を参照してください。

S3 Standard – IA および S3 1 ゾーン – IA の最低ストレージ料金 (30 日分)

S3 Standard – IA および S3 1 ゾーン – IA ストレージクラスには、最低 30 日間のストレージ料金が設定されています。したがって、S3 Standard – IA または S3 1 ゾーン – IA の移行後 30 日以内に S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive の移行が発生する場合、S3 Standard – IA または S3 1 ゾーン – IA の移行および S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive 移行の両方に対して 1 つのライフサイクルルールを指定することはできません。

S3 Standard – IA ストレージから S3 1 ゾーン – IA への移行を指定する場合、同じ 30 日分の最低料金が適用されます。2 つのルールを指定してこれを達成できますが、最小限のストレージ料金が発生します。コストに関する考慮事項の詳細については、「Amazon S3 の料金」を参照してください。

オブジェクトのライフサイクル全体を管理する

オブジェクトの完全なライフサイクルを管理するために、これらの S3 ライフサイクルアクションを組み合わせることができます。たとえば、作成するオブジェクトに、よく定義されたライフサイクルがあるとします。最初に、オブジェクトは 30 日の期間にわたり頻繁にアクセスされます。次に、オブジェクトは最大 90 日間まで、頻繁にアクセスされません。その後は、オブジェクトは不要になるため、アーカイブまたは削除することができます。

このシナリオでは、S3 Intelligent-Tiering、S3 Standard – IA、または S3 1 ゾーン – IA のストレージへの最初の移行アクション、アーカイブのための S3 Glacier ストレージへの別の移行アクション、および失効アクションを指定する S3 ライフサイクルのルールを作成できます。あるストレージクラスから別のストレージクラスにオブジェクトを移動すると、ストレージコストを節約できます。コストに関する考慮事項の詳細については、「Amazon S3 の料金」を参照してください。

S3 Glacier Flexible Retrieval と S3 Glacier Deep Archive ストレージクラスへの移行 (オブジェクトのアーカイブ)

S3 ライフサイクル設定を使用すると、オブジェクトをアーカイブ用に S3 Glacier または S3 Glacier Deep Archive のストレージクラスに移行できます。S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスを選択した場合、オブジェクトは Amazon S3 に維持されます。Amazon S3 Glacier サービスを別途使用して直接アクセスすることはできません。S3 Glacier の一般情報については、Amazon S3 Glacier デベロッパーガイドの「What is Amazon S3 Glacier」を参照してください。

オブジェクトをアーカイブする前に、以下の関連する考慮事項のセクションを確認してください。

一般的な考慮事項

オブジェクトをアーカイブする前に検討する必要のある一般的な考慮事項を以下に示します。

- 暗号化済みオブジェクトは、ストレージクラスの移行プロセス全体を通して暗号化されたままになります。
- S3 Glacier または S3 Glacier Deep Archive ストレージクラスに保存されているオブジェクトはリアルタイムでは利用できません。

  アーカイブされたオブジェクトは Amazon S3 オブジェクトですが、アーカイブされたオブジェクトにアクセスする前に、まずその一時コピーを復元する必要があります。復元されたオブジェクトのコピーは、復元リクエストで指定した期間内のみ利用できます。その後、Amazon S3 によって一時コピーが削除され、オブジェクトは Amazon S3 Glacier Flexible Retrieval にアーカイブされたまま残ります。

  Amazon S3 コンソールを使用して、またはプログラムのコード内で AWS SDK ラッパーライブラリや Amazon S3 REST API を使用して、オブジェクトを復元することができます。詳細については、「アーカイブされたオブジェクトの復元」を参照してください。
- S3 Glacier Flexible Retrieval ストレージクラスに格納されているオブジェクトは、S3 Glacier Deep Archive ストレージクラスにのみ移行できます。

S3 ライフサイクル設定ルールを使用した場合、S3 Glacier から S3 Glacier Deep Archive ストレージクラスにのみオブジェクトのストレージクラスを変換できます。S3 Glacier に格納されているオブジェクトのストレージクラスを S3 Glacier Deep Archive 以外のストレージクラスに変更する場合は、まず復元オペレーションを使用してオブジェクトの一時的なコピーを作成する必要があります。その後、コピーオペレーションを使用して、ストレージクラスとして S3 Standard、S3 Intelligent-Tiering、S3 Standard – IA、S3 1 ゾーン – IA、または低冗長化を指定しているオブジェクトを上書きします。

- S3 Glacier Deep Archive ストレージクラスへのオブジェクトの移行は一方向のみです。

  S3 ライフサイクル設定ルールを使用して、オブジェクトのストレージクラスを S3 Glacier Deep Archive からその他のストレージクラスに変換することはできません。アーカイブされたオブジェクトのストレージクラスをその他のストレージクラスに変更する場合、まず復元オペレーションを使用してオブジェクトの一時コピーを作成する必要があります。その後、コピーオペレーションを使用して、ストレージクラスとして S3 Standard、S3 Intelligent-Tiering、S3 標準 – IA、S3 1 ゾーン – IA、 S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval、または Reduced Redundancy Storage を指定しているオブジェクトを上書きします。

  > **Note**
  >
  > Amazon S3 コンソールでは、S3 Glacier Flexible Retrieval ストレージクラスまたは S3 Glacier Deep Archive ストレージクラス内のオブジェクトに対する、復元したオブジェクトのコピー操作はサポートしていません。このタイプのコピー操作には、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用します。

  S3 Glacier および S3 Glacier Deep Archive ストレージクラスに保存されているオブジェクトは、Amazon S3 でのみ表示され、使用できます。個々の Amazon S3 Glacier サービスから使用することはできません。

  これらは Amazon S3 オブジェクトであるため、Amazon S3 コンソールまたは Amazon S3 API を使用することによってのみアクセスできます。アーカイブされたオブジェクトに、個々の Amazon S3 Glacier コンソールまたは Amazon S3 Glacier API 経由でアクセスすることはできません。

コストに関する考慮事項

アクセス頻度の高いデータを数か月あるいは数年間アーカイブする場合、S3 Glacier および S3 Glacier Deep Archive ストレージクラスを使用する事でストレージコストを削減できます。ただ

し、S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスが適切であることを確認するために、以下の点を考慮してください。

- [ストレージオーバーヘッド料金] - オブジェクトを S3 Glacier または S3 Glacier Deep Archive ストレージクラスに移行すると、そのオブジェクトを管理するメタデータを収容するために、各オブジェクトに対して一定量のストレージが追加されます。
  - S3 Glacier または S3 Glacier Deep Archive にアーカイブされたオブジェクトごとに、Amazon S3 ではオブジェクトの名前とその他のメタデータに 8 KB のストレージを使用します。Amazon S3 でこのメタデータを保存する目的は、ユーザーが Amazon S3 API を使用して、アーカイブされたオブジェクトのリアルタイムのリストを取得できるようにすることです。詳細については、[GET Bucket (List Objects)] を参照してください。この追加のストレージに対しては、S3 Standard 料金が発生します。
  - S3 Glacier または S3 Glacier Deep Archive にアーカイブされるオブジェクトごとに、Amazon S3 はインデックスおよび関連するメタデータ用に 32 KB のストレージを追加します。この追加データは、オブジェクトを特定して復元するのに必要です。この追加ストレージに対しては、S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive のレートが課金されます。

  小さいオブジェクトをアーカイブする場合は、これらのストレージ料金を考慮する必要があります。オーバーヘッドコストを削減するには、多数の小さいオブジェクトを少数の大きいオブジェクトに集約する方法があります。
- [オブジェクトをアーカイブしておく日数] — S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive は長期間のアーカイブを行うソリューションです。最小ストレージ期間は、S3 Glacier ストレージクラスの場合は 90 日、S3 Glacier Deep Archive の場合は 180 日です。Amazon S3 Glacier にアーカイブされているデータの削除は、削除するオブジェクトが最小ストレージ期間より長い期間アーカイブされている場合は無料です。アーカイブされたオブジェクトを最小ストレージ期間以内に削除または上書きする場合は、Amazon S3 によって比例配分された早期削除料金が課金されます。早期削除料金の詳細については、「Amazon S3 Glacier から格納後 90 日未満のオブジェクトを削除するにはいくらかかりますか?」を参照してください。質問については、[Amazon S3 のよくある質問] を参照してください。
- [S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive 移行リクエスト料金] — S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスに移行するオブジェクトごとに 1 つの移行リクエストが作成されます。アーカイブリクエストごとにコストが発生します。多数のオブジェクトを移行する場合は、リクエストのコストを考慮する必要があります。小さなオブジェクト、特に 128KB 未満のオブジェクトを含むオブジェクトを混在させてアーカイブする場合は、ライフサイクルオブジェクトサイズフィルタを使用して移行から小さなオブジェクトを除

外し、リクエストコストを削減することをお勧めします。S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive は、128KB 未満のオブジェクトの移行を自動的にブロックしません。

- [S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive データ復元料金] — S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive は、頻繁にアクセスしないデータの長期アーカイブ用に設計されています。データ復元料金の詳細については、「Amazon S3 Glacier からのデータの復元には、どれだけコストがかかりますか?」を参照してください。質問については、[Amazon S3 のよくある質問] を参照してください。Amazon S3 Glacier からデータを復元する方法については、[アーカイブされたオブジェクトの復元] を参照してください。

S3 ライフサイクル管理を使用して Amazon S3 Glacier にオブジェクトをアーカイブする場合、Amazon S3 はこれらのオブジェクトを非同期的に移行します。S3 ライフサイクル設定ルールによる移行の日付と、物理的な移行の日付の間には、遅延が発生する場合があります。Amazon S3 Glacier の料金は、ルールに指定された移行日に基づいて課金されます。詳細については、Amazon S3 のよくある質問のAmazon S3 Glacier のセクションを参照してください。

Amazon S3 製品詳細ページに、Amazon S3 オブジェクトをアーカイブする場合の料金表情報と計算例が掲載されています。詳細については、次のトピックを参照してください。

- Amazon S3 Glacier にアーカイブされた Amazon S3 オブジェクトのストレージ料金はどのように計算されますか? 質問については、[Amazon S3 のよくある質問] を参照してください。
- Amazon S3 Glacier から格納後 90 日未満のオブジェクトを削除するにはいくらかかりますか? 質問については、[Amazon S3 のよくある質問] を参照してください。
- Amazon S3 Glacier からデータを取り出すのにどれだけのコストがかかりますか? 質問については、[Amazon S3 のよくある質問] を参照してください。
- 標準ストレージクラスと別のストレージクラスのストレージコストを示す Amazon S3 の料金

アーカイブされたオブジェクトの復元

アーカイブされたオブジェクトにはリアルタイムでアクセスできません。まず復元リクエストを開始してから、リクエストで指定した期間中にオブジェクトの一時コピーが利用できるようになるまで待ちます。復元されたオブジェクトの一時コピーの取得後も、オブジェクトのストレージクラスは S3 Glacier または S3 Glacier Deep Archive が保持されます。(HeadObject または GetObject API オペレーションリクエストは、S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive をストレージクラスとして返します)。

> **Note**
>
> アーカイブを復元する場合、アーカイブの費用 (S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive 料金) の他に、一時的に復元されたコピーの費用 (S3 標準ストレージ料金) も発生します。料金については、「Amazon S3 の料金」を参照してください。

オブジェクトのコピーは、プログラムによって、あるいは Amazon S3 コンソールを使用して復元できます。Amazon S3 は、各オブジェクトに対して一度に 1 つの復元リクエストのみを処理します。詳細については、「アーカイブされたオブジェクトの復元」を参照してください。

オブジェクトの有効期限

ライフサイクル設定に基づいて、オブジェクトの存続期間が終了すると、Amazon S3 はバケットの S3 バージョニング状態に応じて自動的にアクションを実行します。

- バージョニングされていないバケット – Amazon S3 はオブジェクトを削除キューに追加して非同期的に削除し、オブジェクトを完全に削除します。
- バージョニングが有効なバケット – 現在のオブジェクトバージョンが削除マーカーではない場合、Amazon S3 は固有のバージョン ID を持つ削除マーカーを追加します。これにより、最新のバージョンが最新以外のバージョンとなり、削除マーカーが最新バージョンになります。
- バージョニングが停止されたバケット — Amazon S3 はバージョン ID として null を使用する削除マーカーを作成します。この削除マーカーは、バージョン階層においてあらゆるオブジェクトバージョンをバージョン ID null に置き換えるため、オブジェクトが効率的に削除されます。

バージョニング対応のバケット (つまり、バージョニングが有効であるかバージョニングが停止されている) の場合、Amazon S3 による Expiration アクションの処理方法を決める考慮事項がいくつかあります。バージョニングが有効なバケットまたはバージョニングが停止されたバケットには、以下が適用されます。

- オブジェクトの有効期限は、オブジェクトの最新のバージョンにのみ適用されます(最新以外のバージョンには影響しません)。
- 1 つ以上のオブジェクトバージョンがあり、削除マーカーが最新バージョンである場合、Amazon S3 はアクションを実行しません。
- 現在のオブジェクトバージョンが唯一のオブジェクトバージョンであり、削除マーカーでもある場合 (「期限切れオブジェクト削除マーカー」とも呼ばれます。すべてのオブジェクトバージョンが削除され、削除マーカーのみ残ります)、Amazon S3 は期限切れのオブジェクト削除マーカーを削

除します。また、expiration アクションを使用して、期限切れオブジェクト削除マーカーをすべて削除するよう Amazon S3 に指示することもできます。例については、「例 7: 期限切れオブジェクト削除マーカーを削除する」を参照してください。

- アクション要素 NoncurrentVersionExpiration を使用すると、Amazon S3 が最新以外のオブジェクトのバージョンを永続的に削除するように指示できます。このように削除されたオブジェクトは復元することはできません。この有効期限切れは、オブジェクトが最新でなくなった以降の特定の日数に基づいて設定できます。日数以外にも、保持する最新以外のバージョンの最大数 (1 ~ 100 の間) を指定することもできます。この値は、Amazon S3 で関連付けられたアクションが実行できるまでに、最新でないバージョンがいくつ必要かを指定します。最新以外のバージョンの最大数を指定するには、Filter 要素も指定する必要があります。Filter 要素を指定しないで最新以外のバージョンの最大数を指定すると、Amazon S3 で InvalidRequest エラーが発生します。アクション要素 NoncurrentVersionExpiration の使用方法の詳細については、「the section called “ライフサイクルアクションを記述する要素”」を参照してください。

詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

> **Important**
>
> S3 ライフサイクル設定に複数のルールがある場合、1 つのオブジェクトが同じ日に複数の S3 ライフサイクルアクションの対象になることがあります。このような場合、Amazon S3 は以下の一般的なルールに従います。
>
> - 完全な削除は、移行より優先されます。
> - 移行は、削除マーカーの作成より優先されます。
> - オブジェクトが S3 Glacier Flexible Retrieval と S3 Standard-IA (または S3 One Zone-IA) 移行の両方の対象になる場合、Amazon S3 は S3 Glacier Flexible Retrieval 移行を選択します。
>
> 例については、「例 5: 重複するフィルター、競合するライフサイクルアクション、Amazon S3 がバージョン管理されていないバケットに対して行うこと」を参照してください。

既存のオブジェクトと新しいオブジェクト

ライフサイクル設定をバケットに追加すると、設定ルールは既存のオブジェクトとそれ以降に追加されるオブジェクトの両方に適用されます。例えば、特定のプレフィックスが付いたオブジェクトが作成から 30 日後に有効期限を迎えるようにする失効アクションを備えたライフサイクル設定ルールを

本日追加すると、Amazon S3 により作成から 30 日以上が経過し、指定されたプレフィックスを持つ既存のオブジェクトがすべて削除キューに追加されます。

> **Important**
>
> バケットポリシーを使用して、S3 ライフサイクルルールによる削除や移行を防ぐことはできません。例えば、バケットポリシーがすべてのプリンシパルのすべてのアクションを拒否する場合でも、S3 ライフサイクル設定は通常どおり機能します。

オブジェクトの有効期限を確認する方法

オブジェクトでスケジュールされている有効期限を確認するには、HeadObject または GetObject の API オペレーションを使用します。これらの API オペレーションでは、オブジェクトがキャッシュ可能でなくなる日時が指定されたレスポンスヘッダーが返されます。

> **Note**
>
> - 有効期限が切れる日と Amazon S3 がオブジェクトを削除する日との間に遅延が生じることがあります。期限切れのオブジェクトに関連付けられている有効期限切れ、またはストレージ期間に対する料金は請求されません。
> - ライフサイクルルールを更新、無効化、または削除する前に、LIST API オペレーション (ListObjectsV2、ListObjectVersions、ListMultipartUploads など) または Amazon S3 インベントリ を使用して、ユースケースに基づき、Amazon S3 が移行済みで対象となるオブジェクトの有効期限が切れていることを確認します。

最小ストレージ期間料金

S3 Standard – IA、または S3 1 ゾーン – IA のストレージにあった期間が 30 日未満のオブジェクトを失効させる S3 ライフサイクル失効ルールを作成すると、30 日間の料金が発生します。S3 Glacier ストレージにあった期間が 90 日未満のオブジェクトを失効させるライフサイクル有効期限ルールを作成すると、90 日間の料金が発生します。S3 Glacier Deep Archive ストレージにあった期間が 180 日未満のオブジェクトを失効させるライフサイクル失効ルールを作成すると、180 日間の料金が発生します。

詳細については、「Amazon S3 の料金」を参照してください。

バケットにライフサイクル設定を設定する

このセクションでは、Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して、バケットに Amazon S3 ライフサイクル設定を指定する方法について説明します。S3 ライフサイクル設定については、「ストレージのライフサイクルの管理」を参照してください。

ライフサイクルルールを使用すると、オブジェクトのライフタイムで Amazon S3 が実行するアクションを定義することができます (他のストレージクラスへのオブジェクトの移行や指定した期間後のオブジェクトのアーカイブや削除など)。

ライフサイクル設定を設定する前に、次の点に注意してください。

ライフサイクル設定の伝播の遅延

バケットに対してライフサイクル設定を追加する場合、新しい、あるいは更新された S3 ライフサイクル設定がすべての Amazon S3 システムに完全に伝達されるまでには、通常、多少のタイムラグがあります。ライフサイクル設定が完全に有効になるまで、数分間程度の遅延を想定してください。この遅延は、S3 ライフサイクル設定を削除するときにも発生することがあります。

移行または有効期限の遅延

ライフサイクルルールが満たされてから、ルールのアクションが完了するまでに遅延が生じる場合があります。例えば、ライフサイクルルールによって一連のオブジェクトの有効期限が 1 月 1 日に切れるとします。有効期限ルールは 1 月 1 日に満たされますが、Amazon S3 は数日後または数週間後までこれらのオブジェクトを実際に削除しない場合があります。この遅延は、S3 ライフサイクルがオブジェクトを移行または有効期限のためのキューに非同期的に入れるために発生します。請求の変更は、通常、アクションが完了していなくても、ライフサイクルルールが満たされた際に適用されます。詳細については、「請求の変更」を参照してください。アクティブなライフサイクルルールによって行われた更新の影響をモニタリングするには、「the section called “ライフサイクルルールによって実行されたアクションをモニタリングするにはどうすればよいですか？”」を参照してください。

ライフサイクルルールの無効化または削除

ライフサイクルルールを無効にする、あるいは削除すると、わずかなタイムラグの後に、Amazon S3 は新しいオブジェクトの削除や移行のスケジューリングを停止します。スケジュール済みのオブジェクトはスケジュール解除され、削除または移行されません。

> **Note**
>
> ライフサイクルルールを更新、無効化、または削除する前に、LIST API オペレーション (ListObjectsV2、ListObjectVersions、ListMultipartUploads など) または Amazon S3 インベントリ を使用して、ユースケースに基づき、Amazon S3 が移行済みで対象となるオブジェクトの有効期限が切れていることを確認します。ライフサイクルルールの更新、無効化、削除に問題がある場合は、「Amazon S3 ライフサイクル問題のトラブルシューティング」を参照してください。

既存のオブジェクトと新しいオブジェクト

ライフサイクル設定をバケットに追加すると、設定ルールは既存のオブジェクトとそれ以降に追加されるオブジェクトの両方に適用されます。例えば、特定のプレフィックスが付いたオブジェクトが作成から 30 日後に有効期限を迎えるようにする失効アクションを備えたライフサイクル設定ルールを本日追加すると、Amazon S3 により作成から 30 日以上が経過し、指定されたプレフィックスを持つ既存のオブジェクトがすべて削除キューに追加されます。

ライフサイクルルールの影響のモニタリング

アクティブなライフサイクルルールによって行われた更新の影響をモニタリングするには、「the section called “ライフサイクルルールによって実行されたアクションをモニタリングするにはどうすればよいですか？”」を参照してください。

請求の変更

ライフサイクル設定のルールが満たされてから、実際にアクションが実行されるまでの間には、遅延が発生する場合があります。ただし、アクションが実行されていなくても、ライフサイクル設定のルールが満たされた段階で、請求にはすぐに変更が反映されます。

たとえば、オブジェクトの有効期限が切れると、オブジェクトがすぐに削除されなくても、そのストレージは課金対象から外されます。同様に、オブジェクトの移行期間を過ぎると、オブジェクトがすぐに S3 Glacier Flexible Retrieval ストレージクラスに移行されていなくても、S3 Glacier Flexible Retrieval ストレージ料金が請求されます。

ただし、S3 Intelligent-Tiering ストレージクラスへのライフサイクルの移行については例外です。オブジェクトが S3 Intelligent-Tiering ストレージクラスに移行するまで、請求の変更は行われません。

複数のルールまたは競合するルール

S3 ライフサイクル設定に複数のルールがある場合、1 つのオブジェクトが同じ日に複数の S3 ライフサイクルアクションの対象になることがあります。このような場合、Amazon S3 は以下の一般的なルールに従います。

- 完全な削除は、移行より優先されます。
- 移行は、削除マーカーの作成より優先されます。
- オブジェクトが S3 Glacier Flexible Retrieval と S3 Standard-IA (または S3 One Zone-IA) 移行の両方の対象になる場合、Amazon S3 は S3 Glacier Flexible Retrieval 移行を選択します。

例については、「例 5: 重複するフィルター、競合するライフサイクルアクション、Amazon S3 がバージョン管理されていないバケットに対して行うこと」を参照してください。

S3 コンソールの使用

共有プレフィックス (共通の文字列から始まるオブジェクト名) またはタグを使用して、バケット内のすべてのオブジェクトまたはオブジェクトのサブセットに対してライフサイクルルールを定義できます。ライフサイクルルールでは、現在のバージョンまたは以前のバージョンのオブジェクトに対して独自のアクションを定義できます。詳細については、次を参照してください:

- ストレージのライフサイクルの管理
- S3 バケットでのバージョニングの使用

ライフサイクルルールを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets (バケット)] リストで、ライフサイクルルールを作成するバケットの名前を選択します。
3. [Management (管理)] タブを選択して、[Create lifecycle rule (ライフサイクルルールを作成する)] を選択します。
4. [Lifecycle rule name (ライフサイクルルール名)] に、ルールの名前を入力します。

   その名前はバケット内で一意である必要があります。
5. ライフサイクルルールのスコープを選択します。

   - 特定のプレフィックスまたはタグを持つすべてのオブジェクトにこのライフサイクルルールを適用するには、[スコープを特定のプレフィックスまたはタグに制限] を選択します。

- スコープをプレフィックスで制限するには、[Prefix (プレフィックス)] にプレフィックスを入力します。
- スコープをタグで制限するには、[Add tag (タグの追加)] を選択してタグのキーと値を入力します。

オブジェクト名のプレフィックスの詳細については、「オブジェクトキー名の作成」を参照してください。オブジェクトロックの詳細については、タグを使用してストレージを分類するをご参照ください。

- このライフサイクルルールをバケット内のすべてのオブジェクトに適用するには、[このルールは、バケット内のすべてのオブジェクトに適用されます] を選択して、[このライフサイクルルールがバケット内のすべてのオブジェクトに適用されることを了承します] を選択します。

6. オブジェクトサイズでルールをフィルタリングするには、[最小オブジェクトサイズを指定]、[最大オブジェクトサイズを指定]、または両方のオプションを選択します。

   - [最小オブジェクトサイズ] または [最大オブジェクトサイズ] の値を指定する際は、0 バイトより大きく、5 TB 未満の範囲の値を指定する必要があります。この値はバイト、KB、MB、または GB で指定できます。
   - 両方の値を指定する際は、最大オブジェクトサイズを最小オブジェクトサイズよりも大きい値に設定する必要があります。

   > **Note**
   >
   > [最小オブジェクトサイズ] フィルターおよび [最大オブジェクトサイズ] フィルターは、指定された値を除外します。例えば、[最小オブジェクトサイズ] で 128 KB のオブジェクトを期限切れにするようにフィルターを設定した場合、128 KB ちょうどのオブジェクトは期限切れになりません。代わりに、このルールは 128 KB より大きいサイズのオブジェクトにのみ適用されます。

7. [Lifecycle rule actions (ライフサイクルルールのアクション)] で、ライフサイクルルールで実行するアクションを以下から選択します。

   - オブジェクトの現在のバージョンをストレージクラス間で移行する
   - オブジェクトの以前のバージョンをストレージクラス間で移動する
   - オブジェクトの現在のバージョンを期限切れにする

> **Note**
>
> S3 バージョニングが有効化されていないバケットの場合、現在のバージョンの有効期限が切れると、Amazon S3 はオブジェクトを完全に削除します。詳細については、「the section called “ライフサイクルのアクションとバケットのバージョニング状態”」を参照してください。

- オブジェクトの以前のバージョンを完全に削除する
- 期限切れの削除マーカーまたは未完了のマルチパートアップロードを削除する

選択したアクションに応じて、異なるオプションが表示されます。

8. オブジェクトの現在のバージョンをストレージクラス間で移行するには、[Transition current versions of objects between storage classes (オブジェクトの現行バージョンをストレージクラス間で移行する)] で以下の操作を行います。

   a. [ストレージクラスへの移行] で、移行するストレージクラスを選択します。利用できる移行のリストについては、「the section called “サポートされているライフサイクル移行”」を参照してください。以下のストレージクラスから選択できます。

      - S3 Standard – IA
      - S3 Intelligent-Tiering
      - S3 1 ゾーン - IA
      - S3 Glacier Flexible Retrieval
      - S3 Glacier Deep Archive

   b. [Days after object creation (オブジェクトが作成されてからの日数)] に、オブジェクトの作成から移行までの日数を入力します。

   ストレージクラスの詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。オブジェクトの現在のバージョンだけ、以前のバージョンだけ、または現在と以前の両方のバージョンについて、移行を定義できます。バージョニングを使用すると、1 つのバケットで複数バージョンのオブジェクトを維持できます。バージョニングの詳細については、「S3 コンソールの使用」を参照してください。

> **Important**
>
> S3 Glacier Flexible Retrieval あるいは Glacier Deep Archive ストレージクラスを選択した場合は、オブジェクトは Amazon S3 に保持されます。Amazon S3 Glacier サービスを別途使用して直接アクセスすることはできません。詳細については、「Amazon S3 ライフサイクルを使用したオブジェクトの移行」を参照してください。

9. ストレージクラス間で最新以外のオブジェクトを移行するには、[Transition noncurrent versions of objects between storage classes] で次のとおり指定します。

   a. [ストレージクラスへの移行] で、移行するストレージクラスを選択します。利用できる移行のリストについては、「the section called “サポートされているライフサイクル移行”」を参照してください。以下のストレージクラスから選択できます。

      - S3 Standard – IA
      - S3 Intelligent-Tiering
      - S3 1 ゾーン - IA
      - S3 Glacier Flexible Retrieval
      - S3 Glacier Deep Archive

   b. [Days after object becomes noncurrent] に、オブジェクトの作成日から移行日までの日数を入力します。

10. オブジェクトの現行バージョンを期限切れにするには、[Expire current versions of objects] (オブジェクトの現行バージョンを期限切れにする) で、[Number of days after object creation] (オブジェクトの作成後の日数) に日数を入力します。

> **Important**
>
> バージョニング非対応のバケットの場合、アクションを実行すると、Amazon S3 はオブジェクトを完全に削除します。ライフサイクルアクションの詳細については、「ライフサイクルアクションを記述する要素」を参照してください。

11. オブジェクトの以前のバージョンを完全に削除するには、[Permanently delete noncurrent versions of objects] (現在のバージョンではないオブジェクトを完全に削除する) で、[Days after objects become noncurrent] (オブジェクトが最新バージョンでなくなってからの日数) に日数を入力します。オプションで、[Number of newer versions to retain] (保持する新しいバージョンの数) に値を入力して、保持する新しいバージョンの数を指定できます。

12. [Delete expired delete markers or incomplete multipart uploads (期限切れの削除マーカーまたは未完了のマルチパートアップロードを削除する)] で、[Delete expired object delete markers (期限切れのオブジェクト削除マーカーを削除する)] と [Delete incomplete multipart uploads (不完全なマルチパートアップロードを削除)] を選択します。次に、マルチパートアップロードの開始から未完了のマルチパートアップロードを終了してクリーンアップするまでの日数を入力します。

    マルチパートアップロードの詳細については、「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。

13. [Create rule] を選択します。

    ルールにエラーがない場合は、Amazon S3 によって有効にされ、[Management (管理)] タブの [Lifecycle rules (ライフサイクルルール)] に表示されます。

AWS CloudFormation テンプレートと例の詳細については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation テンプレートの使用」および「AWS::S3::Bucket」を参照してください。

AWS CLI の使用

次の AWS CLI コマンドを使用して、S3 ライフサイクル設定を管理できます。

- `put-bucket-lifecycle-configuration`
- `get-bucket-lifecycle-configuration`
- `delete-bucket-lifecycle`

AWS CLI をセットアップする手順については、「AWS CLI を使用した Amazon S3 での開発」を参照してください。

Amazon S3 ライフサイクル設定は XML ファイルである点に注意してください。ただし、AWS CLI を使用する場合、XML 形式を指定することはできません。代わりに JSON 形式を指定する必要があります。XML ライフサイクル設定の例と、AWS CLI コマンドで指定できる同等の JSON の設定は、次のとおりです。

次のサンプル S3 ライフサイクル設定を検討してください。

Example 例 1

Example

XML

```
<LifecycleConfiguration>
    <Rule>
        <ID>ExampleRule</ID>
        <Filter>
           <Prefix>documents/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

JSON

```
{
    "Rules": [
        {
            "Filter": {
                "Prefix": "documents/"
            },
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 365,
                    "StorageClass": "GLACIER"
                }
            ],
            "Expiration": {
                "Days": 3650
            },
            "ID": "ExampleRule"
```

```
        }
    ]
        }
```

Example 例 2

Example

XML

```
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Rule>
        <ID>id-1</ID>
        <Expiration>
            <Days>1</Days>
        </Expiration>
        <Filter>
            <And>
                <Prefix>myprefix</Prefix>
                <Tag>
                    <Key>mytagkey1</Key>
                    <Value>mytagvalue1</Value>
                </Tag>
                <Tag>
                    <Key>mytagkey2</Key>
                    <Value>mytagvalue2</Value>
                </Tag>
            </And>
        </Filter>
        <Status>Enabled</Status>
    </Rule>
</LifecycleConfiguration>
```

JSON

```
{
    "Rules": [
        {
            "ID": "id-1",
            "Filter": {
                "And": {
```

```
                    "Prefix": "myprefix",
                    "Tags": [
                        {
                            "Value": "mytagvalue1",
                            "Key": "mytagkey1"
                        },
                        {
                            "Value": "mytagvalue2",
                            "Key": "mytagkey2"
                        }
                    ]
                }
            },
            "Status": "Enabled",
            "Expiration": {
                "Days": 1
            }
        }
    ]
}
```

以下のように put-bucket-lifecycle-configuration をテストできます。

設定をテストする

1. JSON のライフサイクル設定をファイル (*lifecycle.json* など) に保存します。
2. 以下の AWS CLI コマンドを実行して、バケットにライフサイクル設定を指定します。*user input placeholders* を、ユーザー自身の情報に置き換えます。

```
$ aws s3api put-bucket-lifecycle-configuration  \
--bucket DOC-EXAMPLE-BUCKET  \
--lifecycle-configuration file://lifecycle.json
```

3. 検証するには、次のとおり get-bucket-lifecycle-configuration AWS CLI コマンドを使用して S3 ライフサイクル設定を取得します。

```
$ aws s3api get-bucket-lifecycle-configuration  \
--bucket DOC-EXAMPLE-BUCKET
```

4. S3 ライフサイクル設定を削除するには、次のとおり delete-bucket-lifecycle AWS CLI コマンドを使用します。

```
aws s3api delete-bucket-lifecycle \
--bucket DOC-EXAMPLE-BUCKET
```

AWS SDK の使用

Java

AWS SDK for Java を使用して、バケットの S3 ライフサイクル設定を管理できます。S3 ライフサイクル設定の管理の詳細については、「ストレージのライフサイクルの管理」を参照してください。

> **Note**
>
> バケットに S3 ライフサイクル設定を追加すると、Amazon S3 でバケットの現在のライフサイクル設定 (ある場合) が置き換えられます。設定を更新するには、設定を取得して必要な変更を加え、変更済みの設定をバケットに追加します。

以下の例では、AWS SDK for Java を使用してバケットのライフサイクル設定を追加、更新、削除する方法を示します。この例では、次のような処理を実行します。

- ライフサイクル設定をバケットに追加します。
- ライフサイクル設定を取得し、別のルールを追加して設定を更新します。
- バケットに変更済みのライフサイクル設定を追加します。Amazon S3 によって既存の設定が置き換えられます。
- 設定をもう一度取得し、ルールの数を出力して、適切な数のルールがあることを確認します。
- ライフサイクル設定を削除し、削除済みであることを確認するために、設定の取得を再度試行します。

作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketLifecycleConfiguration;
import com.amazonaws.services.s3.model.BucketLifecycleConfiguration.Transition;
import com.amazonaws.services.s3.model.StorageClass;
import com.amazonaws.services.s3.model.Tag;
import com.amazonaws.services.s3.model.lifecycle.LifecycleAndOperator;
import com.amazonaws.services.s3.model.lifecycle.LifecycleFilter;
import com.amazonaws.services.s3.model.lifecycle.LifecyclePrefixPredicate;
import com.amazonaws.services.s3.model.lifecycle.LifecycleTagPredicate;

import java.io.IOException;
import java.util.Arrays;

public class LifecycleConfiguration {

        public static void main(String[] args) throws IOException {
                Regions clientRegion = Regions.DEFAULT_REGION;
                String bucketName = "*** Bucket name ***";

                // Create a rule to archive objects with the "glacierobjects/"
 prefix to Glacier
                // immediately.
                BucketLifecycleConfiguration.Rule rule1 = new
 BucketLifecycleConfiguration.Rule()
                                .withId("Archive immediately rule")
                                .withFilter(new LifecycleFilter(new
 LifecyclePrefixPredicate("glacierobjects/")))
                                .addTransition(new
 Transition().withDays(0).withStorageClass(StorageClass.Glacier))
                                .withStatus(BucketLifecycleConfiguration.ENABLED);

                // Create a rule to transition objects to the Standard-Infrequent
 Access storage
                // class
                // after 30 days, then to Glacier after 365 days. Amazon S3 will
 delete the
                // objects after 3650 days.
                // The rule applies to all objects with the tag "archive" set to
 "true".
                BucketLifecycleConfiguration.Rule rule2 = new
 BucketLifecycleConfiguration.Rule()
                                .withId("Archive and then delete rule")
```

```
                                .withFilter(new LifecycleFilter(new
LifecycleTagPredicate(new Tag("archive", "true"))))
                                .addTransition(new Transition().withDays(30)

.withStorageClass(StorageClass.StandardInfrequentAccess))
                                .addTransition(new
Transition().withDays(365).withStorageClass(StorageClass.Glacier))
                                .withExpirationInDays(3650)
                                .withStatus(BucketLifecycleConfiguration.ENABLED);

                // Add the rules to a new BucketLifecycleConfiguration.
                BucketLifecycleConfiguration configuration = new
BucketLifecycleConfiguration()
                                .withRules(Arrays.asList(rule1, rule2));

                try {
                        AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                                        .withCredentials(new
ProfileCredentialsProvider())
                                        .withRegion(clientRegion)
                                        .build();

                        // Save the configuration.
                        s3Client.setBucketLifecycleConfiguration(bucketName,
configuration);

                        // Retrieve the configuration.
                        configuration =
s3Client.getBucketLifecycleConfiguration(bucketName);

                        // Add a new rule with both a prefix predicate and a tag
predicate.
                        configuration.getRules().add(new
BucketLifecycleConfiguration.Rule().withId("NewRule")
                                        .withFilter(new LifecycleFilter(new
LifecycleAndOperator(
                                                        Arrays.asList(new
LifecyclePrefixPredicate("YearlyDocuments/"),
                                                                        new
LifecycleTagPredicate(new Tag(

   "expire_after",

   "ten_years"))))))
```

```
                                        .withExpirationInDays(3650)

 .withStatus(BucketLifecycleConfiguration.ENABLED));

                        // Save the configuration.
                        s3Client.setBucketLifecycleConfiguration(bucketName,
 configuration);

                        // Retrieve the configuration.
                        configuration =
 s3Client.getBucketLifecycleConfiguration(bucketName);

                        // Verify that the configuration now has three rules.
                        configuration =
 s3Client.getBucketLifecycleConfiguration(bucketName);
                        System.out.println("Expected # of rules = 3; found: " +
 configuration.getRules().size());

                        // Delete the configuration.
                        s3Client.deleteBucketLifecycleConfiguration(bucketName);

                        // Verify that the configuration has been deleted by
 attempting to retrieve it.
                        configuration =
 s3Client.getBucketLifecycleConfiguration(bucketName);
                        String s = (configuration == null) ? "No configuration
 found." : "Configuration found.";
                        System.out.println(s);
                } catch (AmazonServiceException e) {
                        // The call was transmitted successfully, but Amazon S3
 couldn't process
                        // it, so it returned an error response.
                        e.printStackTrace();
                } catch (SdkClientException e) {
                        // Amazon S3 couldn't be contacted for a response, or the
 client
                        // couldn't parse the response from Amazon S3.
                        e.printStackTrace();
                }
        }
}
```

.NET

AWS SDK for .NET を使用して、バケットの S3 ライフサイクル設定を管理できます。ライフサイクル設定の管理の詳細については、「ストレージのライフサイクルの管理」を参照してください。

> **Note**
>
> ライフサイクル設定を追加すると、Amazon S3 で、指定したバケットの既存のライフサイクル設定が置き換えられます。設定を更新するには、まずライフサイクル設定を取得して変更を加え、次に変更済みのライフサイクル設定をバケットに追加する必要があります。

以下の例では、AWS SDK for .NET を使用してバケットのライフサイクル設定を追加、更新、削除する方法を示します。このコード例では、以下を行います。

- ライフサイクル設定をバケットに追加します。
- ライフサイクル設定を取得し、別のルールを追加して設定を更新します。
- バケットに変更済みのライフサイクル設定を追加します。Amazon S3 によって既存のライフサイクル設定が置き換えられます。
- 再び設定を取得し、設定のルール数を出力して適切な数であることを確認します。
- ライフサイクル設定を削除し、削除を確認します。

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class LifecycleTest
    {
        private const string bucketName = "*** bucket name ***";
```

```
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddUpdateDeleteLifecycleConfigAsync().Wait();
        }

        private static async Task AddUpdateDeleteLifecycleConfigAsync()
        {
            try
            {
                var lifeCycleConfiguration = new LifecycleConfiguration()
                {
                    Rules = new List<LifecycleRule>
                        {
                            new LifecycleRule
                            {
                                 Id = "Archive immediately rule",
                                 Filter = new LifecycleFilter()
                                 {
                                     LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                     {
                                         Prefix = "glacierobjects/"
                                     }
                                 },
                                 Status = LifecycleRuleStatus.Enabled,
                                 Transitions = new List<LifecycleTransition>
                                 {
                                  new LifecycleTransition
                                  {
                                       Days = 0,
                                       StorageClass = S3StorageClass.Glacier
                                  }
                                 },
                            },
                            new LifecycleRule
                            {
                                 Id = "Archive and then delete rule",
                                  Filter = new LifecycleFilter()
                                 {
```

```
                                    LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                    {
                                        Prefix = "projectdocs/"
                                    }
                                },
                                Status = LifecycleRuleStatus.Enabled,
                                Transitions = new List<LifecycleTransition>
                                {
                                     new LifecycleTransition
                                     {
                                          Days = 30,
                                          StorageClass =
S3StorageClass.StandardInfrequentAccess
                                     },
                                     new LifecycleTransition
                                     {
                                       Days = 365,
                                       StorageClass = S3StorageClass.Glacier
                                     }
                                },
                                Expiration = new LifecycleRuleExpiration()
                                {
                                      Days = 3650
                                }
                            }
                        }
                };

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Retrieve an existing configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

                // Add a new rule.
                lifeCycleConfiguration.Rules.Add(new LifecycleRule
                {
                    Id = "NewRule",
                    Filter = new LifecycleFilter()
                    {
                        LifecycleFilterPredicate = new LifecyclePrefixPredicate()
                        {
```

```
                            Prefix = "YearlyDocuments/"
                        }
                    },
                    Expiration = new LifecycleRuleExpiration()
                    {
                        Days = 3650
                    }
                });

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Verify that there are now three rules.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);
                Console.WriteLine("Expected # of rulest=3; found:{0}",
lifeCycleConfiguration.Rules.Count);

                // Delete the configuration.
                await RemoveLifecycleConfigAsync(client);

                // Retrieve a nonexistent configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
LifecycleConfiguration configuration)
        {

            PutLifecycleConfigurationRequest request = new
PutLifecycleConfigurationRequest
            {
```

```
                BucketName = bucketName,
                Configuration = configuration
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }

        static async Task<LifecycleConfiguration>
 RetrieveLifecycleConfigAsync(IAmazonS3 client)
        {
            GetLifecycleConfigurationRequest request = new
 GetLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }

        static async Task RemoveLifecycleConfigAsync(IAmazonS3 client)
        {
            DeleteLifecycleConfigurationRequest request = new
 DeleteLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
    }
}
```

Ruby

AWS SDK for Ruby では、クラス AWS::S3::BucketLifecycleConfiguration を使用して、バケットの S3 ライフサイクル設定を管理できます。ライフサイクル設定の管理の詳細については、「ストレージのライフサイクルの管理」を参照してください。

REST API の使用

Amazon Simple Storage Service API リファレンスの以下のセクションでは、S3 ライフサイクル設定に関連する REST API について説明します。

- PutBucketLifecycleConfiguration
- GetBucketLifecycleConfiguration
- DeleteBucketLifecycle

S3 ライフサイクルのトラブルシューティング

S3 ライフサイクルの使用時に発生する可能性がある一般的な問題については、「the section called "ライフサイクル問題のトラブルシューティング"」を参照してください。

ライフサイクルとその他のバケット設定

S3 ライフサイクル設定に加えて、バケットに他の設定を関連付けることができます。このセクションでは、S3 ライフサイクル設定と他のバケット設定の関連について説明します。

ライフサイクルとバージョニング

S3 ライフサイクル設定は、バージョニング対応および非対応のどちらのバケットにも追加できます。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

バケットのバージョニングが有効になっている場合は、最新のオブジェクトバージョン 1 個と、0 個以上の以前のバージョンが維持されます。現在のオブジェクトバージョンと以前のオブジェクトバージョン用に異なるライフサイクルルールを定義できます。

詳細については、「ライフサイクル設定の要素」を参照してください。

> **Important**
>
> S3 ライフサイクル設定に複数のルールがある場合、1 つのオブジェクトが同じ日に複数の S3 ライフサイクルアクションの対象になることがあります。このような場合、Amazon S3 は以下の一般的なルールに従います。
>
> - 完全な削除は、移行より優先されます。
> - 移行は、削除マーカーの作成より優先されます。
> - オブジェクトが S3 Glacier Flexible Retrieval 移行と S3 標準 – IA 移行 (または S3 1 ゾーン – IA 移行) の両方の対象になる場合、Amazon S3 は S3 Glacier Flexible Retrieval の移行を選択します。

例については、「例 5: 重複するフィルター、競合するライフサイクルアクション、Amazon S3 がバージョン管理されていないバケットに対して行うこと」を参照してください。

MFA が有効なバケットでのライフサイクル設定

多要素認証 (MFA) が有効なバケットのライフサイクル設定はサポートされていません。

ライフサイクルとログ記録

Amazon S3 ライフサイクルアクションは、AWS CloudTrail オブジェクトレベルのログ記録ではキャプチャされません。CloudTrail は外部の Amazon S3 エンドポイントに対して行われた API リクエストをキャプチャしますが、S3 ライフサイクルアクションは内部の Amazon S3 エンドポイントを使用して実行されます。S3 バケットで Amazon S3 サーバーアクセスログを有効にして、別のストレージクラスへのオブジェクトの移行やオブジェクトの失効など、S3 ライフサイクル関連のアクションをキャプチャして、完全な削除または論理的な削除を行うことができます。詳細については、「the section called “サーバーアクセスのログ記録”」を参照してください。

バケットでログ記録を有効にしている場合、Amazon S3 サーバーアクセスログで以下のオペレーションの結果がレポートされます。

オペレーションログ	説明
S3.EXPIRE.OBJECT	Amazon S3 は、ライフサイクルの有効期限切れアクションに沿って、オブジェクトを完全に削除します。
S3.CREATE.DELETEMARKER	Amazon S3 は現在のバージョンを論理的に削除し、バージョニングが有効なバケットに削除マーカーを追加します。
S3.TRANSITION_SIA.OBJECT	Amazon S3 はオブジェクトを S3 Standard – IA ストレージクラスに移行しました。
S3.TRANSITION_ZIA.OBJECT	Amazon S3 はオブジェクトを S3 1 ゾーン – IA ストレージクラスに移行しました。

オペレーションログ	説明
S3.TRANSITION_INT.OBJECT	Amazon S3 はオブジェクトを S3 Intelligent-Tiering ストレージクラスに移行します。
S3.TRANSITION_GIR.OBJECT	Amazon S3 は、オブジェクトの S3 Glacier Instant Retrieval ストレージクラスへの移行を開始します。
S3.TRANSITION.OBJECT	Amazon S3 は、オブジェクトの S3 Glacier Flexible Retrieval ストレージクラスへの移行を開始します。
S3.TRANSITION_GDA.OBJECT	Amazon S3 は、オブジェクトの S3 Glacier Deep Archive ストレージクラスへの移行を開始します。
S3.DELETE.UPLOAD	Amazon S3 は不完全なマルチパートアップロードを中止します。

> **Note**
>
> Amazon S3 サーバーアクセスログレコードは通常、ベストエフォートベースで配信され、すべての Amazon S3 リクエストの完全なアカウンティングには使用できません。

S3 ライフサイクルのトラブルシューティング

一般的な問題のトラブルシューティングの詳細については、「Amazon S3 ライフサイクル問題のトラブルシューティング」を参照してください。

詳細情報

- ライフサイクル設定の要素
- S3 Glacier Flexible Retrieval と S3 Glacier Deep Archive ストレージクラスへの移行 (オブジェクトのアーカイブ)
- バケットにライフサイクル設定を設定する

ライフサイクルイベント通知の設定

Amazon S3 イベント通知を設定して、Amazon S3 がオブジェクトを削除したり、S3 ライフサイクルルールに従って別の Amazon S3 ストレージクラスに移行したときに通知を受け取ることができます。

`LifecycleExpiration` イベントタイプを使用すると、Amazon S3 が S3 ライフサイクル設定に基づいてオブジェクトを削除する都度、通知を受け取ることができます。`s3:LifecycleExpiration:Delete` イベントタイプは、バージョン管理されていないバケット内のオブジェクトが削除されたときに通知します。また、S3 ライフサイクル設定によってオブジェクトバージョンが完全に削除された場合にも通知されます。この `s3:LifecycleExpiration:DeleteMarkerCreated` は、バージョニングされているバケット内のオブジェクトの最新バージョンが削除され、S3 ライフサイクルが削除マーカーを作成した場合に通知します。さらなる詳細については、Delete object version を参照してください。

`s3:LifecycleTransition` イベントタイプを使用すると、S3 ライフサイクル設定により、ある Amazon S3 ストレージクラスから別のストレージクラスにオブジェクトが移行された場合に通知を受け取ることができます。

Amazon S3 は、Amazon Simple Notification Service (Amazon SNS) トピック、Amazon Simple Queue Service (Amazon SQS) キュー、または AWS Lambda 関数にイベント通知を発行できます。詳細については、「Amazon S3 イベント通知」を参照してください。

Amazon S3 イベント通知を設定する手順については、Enabling event notifications を参照してください。

次は Amazon S3 が `s3:LifecycleExpiration:Delete` イベントを発行するために送信するメッセージの例です。詳細については、イベントメッセージの構成 を参照してください。

```
{
   "Records":[
      {
         "eventVersion":"2.3",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"LifecycleExpiration:Delete",
         "userIdentity":{
            "principalId":"s3.amazonaws.com"
         },
         "requestParameters":{
```

```
            "sourceIPAddress":"s3.amazonaws.com"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"example-s3-bucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::example-s3-bucket"
            },
            "object":{
               "key":"expiration/delete",
               "sequencer":"0055AED6DCD90281E5",
            }
         }
      }
   ]
}
```

s3:LifecycleTransition イベントを公開するために Amazon S3 が送信するメッセージには、次の情報も含まれます。

```
"lifecycleEventData":{
    "transitionEventData": {
        "destinationStorageClass": the destination storage class for the object
    }
}
```

ライフサイクル設定の要素

トピック

- ID 要素
- Status 要素
- Filter 要素

- ライフサイクルアクションを記述する要素

Amazon S3 ライフサイクル設定は、単一または複数のライフサイクルルールで構成される XML として指定します。

```
<LifecycleConfiguration>
    <Rule>
         ...
    </Rule>
    <Rule>
         ...
    </Rule>
</LifecycleConfiguration>
```

各ルールには次が含まれます。

- ルール ID とルールが有効か無効かを示すステータスを含むルールメタデータ。ルールが無効な場合、Amazon S3 はルールで指定されているアクションを実行しません。
- ルールが適用されるオブジェクトを特定するフィルター。フィルターを指定するには、オブジェクトサイズ、オブジェクトキープレフィックス、単一または複数のオブジェクトタグ、またはフィルターの組み合わせを使用できます。
- 指定されたアクションを Amazon S3 に実行させる場合、オブジェクトの有効期間内の日付または期間が指定された 1 つ以上の移行または失効アクション。

以下のセクションでは、S3 ライフサイクル設定における XML 要素について説明します。設定例については、「S3 ライフサイクル設定の例」を参照してください。

ID 要素

S3 ライフサイクル設定には最大 1,000 個のルールを記述できます。この制限は調整できません。<ID> 要素は、ルールを一意に識別します。ID の長さは、255 文字に制限されています。

Status 要素

<Status> 要素の値は、Enabled または Disabled のいずれかです。ルールが無効な場合、Amazon S3 はルールで定義されたアクションを実行しません。

Filter 要素

ライフサイクルルールは、ライフサイクルルールで指定した <Filter> 要素に基づいて、バケット内のすべてのオブジェクトまたはサブセットに適用できます。

オブジェクトは、キープレフィックス、オブジェクトタグ、またはそれらの組み合わせ (この場合、Amazon S3 は論理 AND を使用してフィルタを組み合わせます) により、オブジェクトをフィルタできます。次の例を考えます。

- キープレフィックスを使用したフィルターの指定 – この例は、キー名のプレフィックス (logs/) に基づいてオブジェクトのサブセットに適用される S3 ライフサイクルルールを示しています。例えば、ライフサイクルルールはオブジェクト logs/mylog.txt、logs/temp1.txt、logs/test.txt に適用されます。このルールは、オブジェクト example.jpg には適用されません。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Prefix>logs/</Prefix>
        </Filter>
        transition/expiration actions
          ...
    </Rule>
    ...
</LifecycleConfiguration>
```

異なるキー名のプレフィックスに基づいてオブジェクトのサブセットにライフサイクルアクションを適用する場合は、別のルールを指定します。各ルールでは、プレフィックスベースのフィルタを指定します。例えば、キープレフィックスが projectA/ と projectB/ のオブジェクトのライフサイクルアクションを作成するには、次のとおり 2 つのルールを指定します。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Prefix>projectA/</Prefix>
        </Filter>
        transition/expiration actions
          ...
    </Rule>

    <Rule>
        <Filter>
```

```
            <Prefix>projectB/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>
</LifecycleConfiguration>
```

オブジェクトキーの詳細については、「オブジェクトキー名の作成」を参照してください。

- オブジェクトタグに基づくフィルターの指定 – 次の例では、ライフサイクルルールはタグ (*key*) と値 (*value*) に基づいてフィルターを指定します。この場合、ルールは特定のタグを持つオブジェクトのサブセットにのみ適用されます。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Tag>
              <Key>key</Key>
              <Value>value</Value>
           </Tag>
        </Filter>
        transition/expiration actions
        ...
    </Rule>
</LifecycleConfiguration>
```

複数のタグに基づいてフィルタを指定できます。次の例で示されるとおり、<And> 要素でタグを囲む必要があります。ルールは、2 つのタグ (特定のタグキーと値を持つ) が付与されたオブジェクトでライフサイクルアクションを実行するように Amazon S3 に指示します。

```
<LifecycleConfiguration>
    <Rule>
      <Filter>
         <And>
            <Tag>
               <Key>key1</Key>
               <Value>value1</Value>
            </Tag>
            <Tag>
               <Key>key2</Key>
               <Value>value2</Value>
            </Tag>
```

```
            ...
          </And>
      </Filter>
      transition/expiration actions
    </Rule>
</Lifecycle>
```

ライフサイクルルールは、指定された両方のタグが付与されたオブジェクトに適用されます。Amazon S3 は論理 AND を実行します。次の点に注意してください。

- 各タグは、キーと値の両方で正確に一致する必要があります。<Key> 要素のみを指定し、<Value> 要素を指定しない場合、ルールはタグキーと一致し、値が指定されていないオブジェクトにのみ適用されます。
- ルールは、ルールで指定されたすべてのタグが付与されたオブジェクトのサブセットに適用されます。オブジェクトに追加のタグが指定されていても、ルールは引き続き適用されます。

> **Note**
>
> フィルタで複数のタグを指定する場合、各タグキーは一意である必要があります。

- プレフィックスと単一または複数のタグの両方に基づくフィルターの指定 – ライフサイクルルールでは、キープレフィックスと単一または複数のタグの両方に基づいてフィルターを指定できます。繰り返しになりますが、次に示すとおり、これらすべてを <And> 要素で囲む必要があります。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
          <And>
             <Prefix>key-prefix</Prefix>
             <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
             </Tag>
             <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
             </Tag>
              ...
          </And>
        </Filter>
```

```
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

Amazon S3 は、論理 AND を使用してこのようなフィルターを組み合わせます。つまり、ルールは指定したキープレフィックスと特定のタグを持つオブジェクトのサブセットに適用されます。フィルタのプレフィックスは最大 1 個、タグは 0 個以上にします。

- 空のフィルタ を指定できます。その場合、ルールはオブジェクト内のすべてのバケットに適用されます。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

- [object size] によりルールをフィルタリングするには、最小サイズ、(ObjectSizeGreaterThan) または最大サイズ (ObjectSizeLessThan) を指定するか、またはオブジェクトのサイズの範囲を指定できます。

  オブジェクトサイズの値はバイト単位です。フィルターの最大サイズは 5 TB です。ストレージクラスによっては、オブジェクトサイズに最小制限があります。詳細については、「Amazon S3 ストレージクラスの比較」を参照してください。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

> **Note**
>
> ObjectSizeGreaterThan フィルターおよび ObjectSizeLessThan フィルターは、指定された値を除外します。例えば、S3 Standard ストレージクラスから S3 Standard-IA ストレージクラスに移動するためにオブジェクトのサイズを 128 KB から 1024 KB に設定した場合、1024 KB ちょうどのオブジェクトと 128 KB ちょうどのオブジェクトは S3 Standard-IA に移動されません。代わりに、このルールは 128 KB より大きく 1024 KB よりも小さいサイズのオブジェクトにのみ適用されます。

オブジェクトサイズの範囲を指定する場合は、ObjectSizeGreaterThan 整数は ObjectSizeLessThan 値未満にする必要があります。複数のフィルタを使用する場合は、<And> エレメントのフィルタを包む必要があります。次の例は、500 バイトから 64,000 バイトの範囲でオブジェクトを指定する方法を示しています。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <And>
                <Prefix>key-prefix</Prefix>
                <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
                <ObjectSizeLessThan>64000</ObjectSizeLessThan>
            </And>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

ライフサイクルアクションを記述する要素

次の定義済みのアクションのうち 1 つ以上を S3 ライフサイクルルールに指定することで、オブジェクトの有効期間中に特定のアクションを実行するように Amazon S3 に指示することができます。これらのアクションの効果は、バケットのバージョニング状態によって異なります。

- **Transition** アクション要素 – オブジェクトを別のストレージクラスに移行するには、Transition アクションを指定します。オブジェクトの移行の詳細については、「サポート

されている移行と関連する制約」を参照してください。オブジェクトの有効期間内の指定の日付または期間に到達すると、Amazon S3 は移行を実行します。

バージョニング対応のバケット (バージョニングが有効なバゲットまたはバージョニングが停止されたバケット) の場合、`Transition` アクションは最新のオブジェクトバージョンに適用されます。以前のバージョンを管理する場合、Amazon S3 は `NoncurrentVersionTransition` アクション (以下で説明されます) を定義します。

- **`Expiration` アクション要素** – `Expiration` アクションは、ルールで特定されたオブジェクトを期限切れにします。これは、Amazon S3 のいずれかのストレージクラスの該当するオブジェクトに適用されます。ストレージクラスの詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。Amazon S3 により有効期限切れになったオブジェクトはいずれも使用できなくなります。オブジェクトが完全に削除されるかどうかは、バケットのバージョニング状態によって異なります。
  - バージョニング非対応のバケット – `Expiration` アクションを実行すると、Amazon S3 はオブジェクトを完全に削除します。
  - バージョニング対応バケット – バージョニング対応バケット (つまり、バージョニングが有効であるかバージョニングが停止されている) の場合、Amazon S3 による `Expiration` アクションの処理方法を決める考慮事項がいくつかあります。バージョニングが有効なバケットまたはバージョニングが停止されたバケットには、以下が適用されます。
    - `Expiration` アクションは、最新のバージョンにのみ適用されます (最新以外のバージョンには影響しません)。
    - 1 つ以上のオブジェクトバージョンがあり、削除マーカーが最新バージョンである場合、Amazon S3 はアクションを実行しません。
    - 現在のオブジェクトバージョンが唯一のオブジェクトバージョンであり、削除マーカーでもある場合 (「期限切れオブジェクト削除マーカー」とも呼ばれます。すべてのオブジェクトバージョンが削除され、削除マーカーのみ残ります)、Amazon S3 は期限切れのオブジェクト削除マーカーを削除します。また、expiration アクションを使用して、期限切れオブジェクト削除マーカーをすべて削除するよう Amazon S3 に指示することもできます。例については、「例 7: 期限切れオブジェクト削除マーカーを削除する」を参照してください。

    詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

    有効期限を管理するために Amazon S3 をセットアップする場合は、次の点も考慮します。

    - バージョニングが有効なバケット

現在のオブジェクトバージョンが削除マーカーではない場合、Amazon S3 は固有のバージョン ID を持つ削除マーカーを追加します。これにより、最新のバージョンが最新以外のバージョンとなり、削除マーカーが最新バージョンになります。

- バージョニングが停止されたバケット

  バージョニングを停止したバケットでは、有効期限切れアクションにより Amazon S3 がバージョン ID として null を使用する削除マーカーを作成します。この削除マーカーは、バージョン階層においてあらゆるオブジェクトバージョンをバージョン ID null に置き換えるため、オブジェクトが効率的に削除されます。

さらに、Amazon S3 はバージョン対応のバケット (つまり、バージョニングが有効なバケットとバージョニングが停止されているバケット) で以前のオブジェクトバージョンを管理するために使用できる次のアクションを提供します。

- **NoncurrentVersionTransition** アクション要素 – このアクションを使用して、Amazon S3 がオブジェクトを指定したストレージクラスに移行するタイミングを指定します。この有効期限切れは、オブジェクトが最新でなくなった以降の特定の日数に基づいて設定できます。日数以外にも、保持する最新以外のバージョンの最大数 (1 ~ 100 の間) を指定することもできます。この値は、Amazon S3 で関連付けられたアクションが実行できるまでに、最新でないバージョンがいくつ必要かを決定します。Amazon S3 は、保存すると指定された数を超える最新でないバージョンをすべて移行します。

  最新以外のバージョンの最大数を指定するには、Filter 要素も指定する必要があります。Filter 要素を指定しないで最新以外のバージョンの最大数を指定すると、Amazon S3 で InvalidRequest エラーが発生します。

  オブジェクトの移行の詳細については、「サポートされている移行と関連する制約」を参照してください。NoncurrentVersionTransition アクション中の日数を指定した日付を Amazon S3 が計算する方法の詳細については、ライフサイクルルール: オブジェクトの存在時間に基づく を参照してください。

- **NoncurrentVersionExpiration** アクション要素 – Amazon S3 に最新以外のオブジェクトのバージョンを完全に削除することを指示するには、このアクションを使用します。このように削除されたオブジェクトは復元することはできません。この有効期限切れは、オブジェクトが最新でなくなった以降の特定の日数に基づいて設定できます。日数以外にも、保持する最新以外のバージョンの最大数 (1 ~ 100 の間) を指定することもできます。この値は、Amazon S3 で関連付けられた

アクションが実行できるまでに、最新でないバージョンがいくつ必要かを指定します。Amazon S3 は、保存すると指定された数を超える最新でないバージョンをすべて完全に削除します。

最新以外のバージョンの最大数を指定するには、`Filter` 要素も指定する必要があります。`Filter` 要素を指定しないで最新以外のバージョンの最大数を指定すると、Amazon S3 で `InvalidRequest` エラーが発生します。

以前のオブジェクトをこのように遅れて削除すると、誤って削除または上書きを修正する必要がある場合に役立ちます。たとえば、最新以外のバージョンになってから 5 日後に削除する有効期間ルールを設定できます。例えば、2014 年 1 月 1 日 10:30 AM (UTC) に、`photo.gif` (バージョン ID 111111) というオブジェクトを作成したとします。2014 年 1 月 2 日 11:30 AM (UTC) に、`photo.gif` (バージョン ID 111111) を誤って削除したとします。これにより、新しいバージョン ID (バージョン ID 4857693 など) の削除マーカーが作成されます。この場合、完全な削除が行われるまでの 5 日間は、元のバージョンの `photo.gif` (バージョン ID 111111) を復元することができます。2014 年 1 月 8 日 00:00 (UTC) に、有効期限切れのライフサイクルルールが実行され、`photo.gif` (バージョン ID 111111) は完全に削除されます。これは、このバージョンが最新バージョンでなくなってから 5 日後です。

`NoncurrentVersionExpiration` アクションの日数を指定した日付を Amazon S3 が計算する方法の詳細については、ライフサイクルルール: オブジェクトの存在時間に基づく を参照してください。

> **Note**
>
> オブジェクトの有効期限切れを決定するライフサイクル設定では、不完全なマルチパートアップロードは削除されません。不完全なマルチパートアップロードを削除するには、`AbortIncompleteMultipartUpload` というライフサイクル設定アクションを使用する必要があります。このアクションについては、このセクションの後半で説明します。

この移行アクションと有効期限切れアクション以外にも、次のライフサイクル設定アクションを使用して、不完全なマルチパートアップロードを停止するか、有効期限切れのオブジェクト削除マーカーを削除するように Amazon S3 に指示できます。

- **`AbortIncompleteMultipartUpload` アクション要素** – この要素を使用すると、マルチパートアップロードの進行を許可する最大時間 (日数) を指定できます。対象となるマルチパートアップロード (ライフサイクルルールで指定されているキー名の `prefix` で特定) が所定の期間内に正常

に完了していない場合、Amazon S3 は不完全なマルチパートアップロードを中止します。詳細については、「マルチパートアップロードの中止」を参照してください。

> **Note**
>
> このライフサイクルアクションは、オブジェクトタグを使用するフィルターが含まれるルールでは指定できません。

- **`ExpiredObjectDeleteMarker`** アクション要素 – バージョニングが有効になっているバケットでは、最新以外のバージョンが 0 の削除マーカーは、期限切れオブジェクト削除マーカーと呼ばれます。このライフサイクルアクションを使用して、期限切れオブジェクト削除マーカーを削除するように S3 に指示できます。例については、「例 7: 期限切れオブジェクト削除マーカーを削除する」を参照してください。

> **Note**
>
> このライフサイクルアクションは、オブジェクトタグを使用するフィルターが含まれるルールでは指定できません。

オブジェクトが最新でなくなってからの期間を Amazon S3 が計算する方法

バージョニングが有効なバケットで、オブジェクトの複数のバージョンを所持することができます。最新バージョンが常に 1 個、最新以外のバージョンが 0 個以上存在します。オブジェクトをアップロードするたびに、最新バージョンは以前のバージョンとして保持され、新しく追加されたバージョンがその代わりに最新バージョンになります。オブジェクトが最新でなくなってからの日数を知るために、Amazon S3 はその次の最新バージョンがいつ作成されたかを調べます。Amazon S3 は、オブジェクトの次の最新バージョンが作成されてからの日数を、オブジェクトが最新でなくなってからの日数として使用します。

> **S3 ライフサイクル設定を使用する場合のオブジェクトの以前のバージョンの復元**
>
> 「以前のバージョンの復元」で説明したとおり、次の 3 つの方法のいずれかを使用して、オブジェクトの以前のバージョンを取得できます。
>
> - 方法 1 – オブジェクトの最新以外のバージョンを同じバケットにコピーします。コピーされたオブジェクトはそのオブジェクトの最新バージョンになり、すべてのオブジェクトバージョンが維持されます。

- 方法 2 – オブジェクトの現在のバージョンを完全に削除します。最新のオブジェクトバージョンを削除すると、結果として、以前のバージョンがそのオブジェクトの最新バージョンになります。

バージョニングが有効になっているバケットで S3 ライフサイクル設定ルールを使用している場合は、ベストプラクティスとして、方法 1 を使用することをお勧めします。
S3 ライフサイクルは、結果整合性モデル下で動作します。最新バージョンが完全に削除されても、変更がすべての Amazon S3 システムに反映されるまで消えない場合があります。(このため、Amazon S3 はこのような削除を一時的に認識しない場合があります)。一方で、以前のオブジェクトを有効期限切れにするように設定したライフサイクルルールは、復元したいオブジェクトを含む以前のオブジェクトを完全に削除する可能性があります。つまり、方法 1 でお勧めしたとおり、代替方法として古いバージョンをコピーする方が安全です。

ライフサイクルのアクションとバケットのバージョニング状態

ライフサイクルルール: オブジェクトの存在時間に基づく

Amazon S3 が指定したアクションを実行できる期間は、オブジェクトの作成 (または変更) からの日数で指定できます。

日数を S3 ライフサイクル設定の `Transition` アクションおよび `Expiration` アクションで指定する場合は、以下の点に注意してください。

- 指定する値は、オブジェクトの作成からアクションが発生するまでの日数です。
- Amazon S3 は、ルールで指定された日数をオブジェクト作成時に加え、翌日の UTC 午前 0 時に対してその結果の時間を四捨五入して時間を算出します。例えば、あるオブジェクトが 2014 年 1 月 15 日 10:30 AM (UTC) に作成され、移行ルールで 3 日と指定した場合、オブジェクトの移行日は 2014 年 1 月 19 日 00:00 (UTC) となります。

Note

Amazon S3 は、各オブジェクトの最終更新日のみを維持します。例えば、Amazon S3 コンソールでは、オブジェクトの [プロパティ] ペインに [最終更新日時] が表示されています。新しいオブジェクトを最初に作成する際は、この日付はオブジェクトの作成日を表示します。オブジェクトを置き換えた場合、日付はそれに応じて変わります。つまり、オブジェクトの作成日と [最終更新日時] は同期しています。

日数をライフサイクル設定の NoncurrentVersionTransition アクションおよび NoncurrentVersionExpiration アクションで指定する場合は、以下の点に注意してください。

- 指定する値は、オブジェクトのバージョンが最新ではなくなった日時 (つまり、オブジェクトが上書きまたは削除された日時) から、Amazon S3 が指定したオブジェクトに対してアクションを実行するまでの日数です。
- Amazon S3 は、オブジェクトの次の最新バージョンが作成された時刻にルールで指定された日数を加算して、その結果の時刻を四捨五入して翌日の UTC 午前 0 時に対して加えて時刻を算出します。例えば、バケット内に 2014 年 1 月 1 日の 10:30 AM (UTC) に作成されたオブジェクトの最新バージョンがあるとします。この最新バージョンを置き換えるオブジェクトの新しいバージョンが 2014 年 1 月 15 日の 10:30 AM (UTC) に作成され、移行ルールで 3 日を指定した場合、オブジェクトの移行日は 2014 年 1 月 19 日 00:00 (UTC) と算出されます。

ライフサイクルルール: 特定の日付に基づく

S3 ライフサイクルルールでアクションを指定する際に、Amazon S3 がアクションを実行する日付を指定できます。特定の日付になると、Amazon S3 は (フィルタ条件に基づいて) すべての対象オブジェクトにアクションに適用します。

過去の日付を使用して S3 ライフサイクルアクションを指定すると、すべての対象オブジェクトは、直ちにそのライフサイクルアクションの対象となります。

> **Important**
>
> 日付ベースのアクションは、1 回限りのアクションではありません。Amazon S3 は、日付が過ぎた後も、ルールのステータスが Enabled である限り、継続して日付ベースのアクションを適用します。
> 例えば、すべてのオブジェクトを削除する日付ベースの Expiration アクションを指定するとします (このルールにはフィルターが指定されていないと仮定します)。Amazon S3 は特定の日付に、バケットのすべてのオブジェクトを期限切れにします。Amazon S3 は、バケット内に作成した新しいオブジェクトも続けて有効期限切れにします。ライフサイクルアクションを停止するには、ライフサイクルルールからアクションを削除するか、ルールを無効にするか、ライフサイクル設定からルールを削除する必要があります。

日付の値は、ISO 8601 形式に従う必要があります。この時刻は常に午前 0 時 (UTC) となります。

> **Note**
>
> Amazon S3 コンソールを使用して日付ベースのライフサイクルルールを作成することはできません。ただし、このようなルールを表示、無効化、または削除することはできます。

S3 ライフサイクル設定の例

このセクションでは、S3 ライフサイクル設定の例を示します。それぞれの例では、各サンプルシナリオで XML をどのように指定するかを示します。

トピック

例 1: フィルタを指定する

各 S3 ライフサイクルルールには、S3 ライフサイクルルール適用先バケット内のオブジェクトのサブセットを識別するのに使用できるフィルターが含まれます。以下の S3 ライフサイクル設定は、フィルタを指定する方法の例です。

- この S3 ライフサイクル設定ルールでは、フィルターはキープレフィックス (`tax/`) を指定しています。したがって、ルールは、キー名プレフィックスが `tax/` のオブジェクト (`tax/doc1.txt`、`tax/doc2.txt` など) に適用されます。

  このルールは、Amazon S3 に以下のことを命じる 2 つのアクションを指定します。

  - 作成されてから 365 日後にオブジェクトを S3 Glacier Flexible Retrieval ストレージクラスに移行します。

- 作成から 3,650 日 (10 年) 後にオブジェクトを削除する (Expiration アクション)。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Transition and Expiration Rule</ID>
    <Filter>
       <Prefix>tax/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

オブジェクトの経過日数を作成後の日数で指定する代わりに、各アクションの日付を指定できます。ただし、Date と Days の両方を同じルールで使用することはできません。

- バケット内のすべてのオブジェクトに S3 ライフサイクルルールを適用する場合は、空のプレフィックスを指定します。以下の設定では、作成から 0 日後にオブジェクトを S3 Glacier Flexible Retrieval ストレージクラスに移行するよう Amazon S3 に指示する Transition アクションをルールとして指定しています。このルールは、オブジェクトが作成後の 午前 0 時 (UTC) に S3 Glacier Flexible Retrieval へのアーカイブ対象となることを意味します。ライフサイクルの制約の詳細については、「制約」を参照してください。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Archive all object same-day upon creation</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
```

```
</LifecycleConfiguration>
```

- フィルタ内にゼロまたは 1 個のキー名のプレフィックスと 0 個以上のオブジェクトタグを指定できます。以下のコード例では、キープレフィックスが tax/ のオブジェクトのサブセットと、指定したキーと値の 2 つのタグを含むオブジェクトに、S3 ライフサイクルルールを適用しています。複数のフィルターを指定するときは、次に示すように <And> 要素を含める必要があることに注意してください (Amazon S3 では論理 AND を適用して、指定されたフィルター条件を結合します)。

```
...
<Filter>
   <And>
      <Prefix>tax/</Prefix>
      <Tag>
         <Key>key1</Key>
         <Value>value1</Value>
      </Tag>
      <Tag>
         <Key>key2</Key>
         <Value>value2</Value>
      </Tag>
    </And>
</Filter>
...
```

- オブジェクトのフィルタはタグに基づいてのみ行うことができます。たとえば、以下の S3 ライフサイクルルールは、2 つのタグが指定されている (プレフィックスは指定されていない) オブジェクトに適用されます。

```
...
<Filter>
   <And>
      <Tag>
         <Key>key1</Key>
         <Value>value1</Value>
      </Tag>
      <Tag>
         <Key>key2</Key>
         <Value>value2</Value>
      </Tag>
    </And>
</Filter>
```

```
...
```

Important

S3 ライフサイクル設定に複数のルールがある場合、1 つのオブジェクトが同じ日に複数の S3 ライフサイクルアクションの対象になることがあります。このような場合、Amazon S3 は以下の一般的なルールに従います。

- 完全な削除は、移行より優先されます。
- 移行は、削除マーカーの作成より優先されます。
- オブジェクトが S3 Glacier Flexible Retrieval と S3 Standard-IA (または S3 One Zone-IA) 移行の両方の対象になる場合、Amazon S3 は S3 Glacier Flexible Retrieval 移行を選択します。

例については、「例 5: 重複するフィルター、競合するライフサイクルアクション、Amazon S3 がバージョン管理されていないバケットに対して行うこと」を参照してください。

例 2: ライフサイクルルールを無効にする

S3 ライフサイクルルールは一時的に無効にできます。以下の S3 ライフサイクル設定では、2 つのルールを指定しています。

- ルール 1 は、`logs/` プレフィックスを持つオブジェクトを作成直後に S3 Glacier Flexible Retrieval ストレージクラスに移行するよう、Amazon S3 に指示します。
- ルール 2 は、`documents/` プレフィックスを持つオブジェクトを作成直後に S3 Glacier Flexible Retrieval ストレージクラスに移行するよう、Amazon S3 に指示します。

設定では、ルール 1 は有効で、ルール 2 は無効です。Amazon S3 は無効なルールを無視します。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
```

```
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule2</ID>
    <Filter>
      <Prefix>documents/</Prefix>
    </Filter>
    <Status>Disabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

例 3: オブジェクトの有効期間全体にわたってストレージクラスの層を下げる

この例では、S3 ライフサイクル設定を使用して、有効期間全体にわたってオブジェクトのストレージクラスの階層を下げます。層を下げると、ストレージコストを削減できます。料金に関する詳細については、「Amazon S3 の料金」を参照してください。

次の S3 ライフサイクル設定では、キー名プレフィックス `logs/` が付いたオブジェクトに適用されるルールを指定します。このルールは、次のアクションを指定します。

- 2 つの移行アクション:
  - 作成されてから 30 日後にオブジェクトを S3 Standard – IA ストレージクラスに移行します。
  - 作成されてから 90 日後にオブジェクトを S3 Glacier Flexible Retrieval ストレージクラスに移行します。
- 作成されてから 1 年後にオブジェクトを削除するよう Amazon S3 に指示する 1 つの失効アクション。

```
<LifecycleConfiguration>
  <Rule>
    <ID>example-id</ID>
    <Filter>
       <Prefix>logs/</Prefix>
```

```
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Transition>
      <Days>90</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Note

すべてのアクションがオブジェクトの同じセットに適用される場合 (フィルターによって識別)、すべての S3 ライフサイクルアクションを記述する 1 つのルールを使用できます。それ以外の場合、それぞれが異なるフィルタを指定する複数のルールを追加できます。

Important

S3 ライフサイクル設定に複数のルールがある場合、1 つのオブジェクトが同じ日に複数の S3 ライフサイクルアクションの対象になることがあります。このような場合、Amazon S3 は以下の一般的なルールに従います。

- 完全な削除は、移行より優先されます。
- 移行は、削除マーカーの作成より優先されます。
- オブジェクトが S3 Glacier Flexible Retrieval と S3 Standard-IA (または S3 One Zone-IA) 移行の両方の対象になる場合、Amazon S3 は S3 Glacier Flexible Retrieval 移行を選択します。

例については、「例 5: 重複するフィルター、競合するライフサイクルアクション、Amazon S3 がバージョン管理されていないバケットに対して行うこと」を参照してください。

例 4: 複数のルールを指定する

オブジェクトごとに異なる S3 ライフサイクルアクションが必要な場合、複数のルールを指定できます。以下の S3 ライフサイクル設定には 2 つのルールがあります。

- ルール 1 は、キー名プレフィックス `classA/` が付いたオブジェクトに適用されます。作成されてから 1 年後にオブジェクトを S3 Glacier Flexible Retrieval ストレージクラスに移行し、作成されてから 10 年後にそれらのオブジェクトを期限切れにするよう、Amazon S3 に指示します。
- ルール 2 は、キー名プレフィックス `classB/` が付いたオブジェクトに適用されます。作成されてから 90 日後にオブジェクトを S3 Standard – IA ストレージクラスに移行し、作成されてから 1 年後に削除するよう Amazon S3 に指示します。

```
<LifecycleConfiguration>
    <Rule>
        <ID>ClassADocRule</ID>
        <Filter>
           <Prefix>classA/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
    <Rule>
        <ID>ClassBDocRule</ID>
        <Filter>
            <Prefix>classB/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>90</Days>
           <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <Expiration>
             <Days>365</Days>
        </Expiration>
```

```
    </Rule>
</LifecycleConfiguration>
```

> **Important**
>
> S3 ライフサイクル設定に複数のルールがある場合、1 つのオブジェクトが同じ日に複数の S3 ライフサイクルアクションの対象になることがあります。このような場合、Amazon S3 は以下の一般的なルールに従います。
>
> - 完全な削除は、移行より優先されます。
> - 移行は、削除マーカーの作成より優先されます。
> - オブジェクトが S3 Glacier Flexible Retrieval と S3 Standard-IA (または S3 One Zone-IA) 移行の両方の対象になる場合、Amazon S3 は S3 Glacier Flexible Retrieval 移行を選択します。
>
> 例については、「例 5: 重複するフィルター、競合するライフサイクルアクション、Amazon S3 がバージョン管理されていないバケットに対して行うこと」を参照してください。

例 5: 重複するフィルター、競合するライフサイクルアクション、Amazon S3 がバージョン管理されていないバケットに対して行うこと

重複するプレフィックスまたはアクションを指定する S3 ライフサイクル設定を指定する場合があります。

一般的に、S3 ライフサイクルはコストに合わせて最適化されます。たとえば、2 つの有効期限ポリシーが重複している場合は、短い有効期限ポリシーが適用されるため、データが予想よりも長く保存されることはありません。同様に、2 つの移行ポリシーが重複している場合は、S3 ライフサイクルによってオブジェクトが低コストのストレージクラスに移行されます。

いずれの場合でも、S3 ライフサイクルによって、最も安価なパスが選択されます。この一般的なルールの例外として、S3 Intelligent-Tiering ストレージクラスの場合があります。S3 Intelligent-Tiering は、S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive ストレージクラスを除くどのストレージクラスよりも S3 ライフサイクルで優先されます。

次の例では、競合の可能性を Amazon S3 でどのように解決するかを示します。

Example 1: 重複したプレフィックス (競合なし)

以下の設定例には、次のように重複するプレフィックスを指定する 2 つのルールがあります。

- 1 番目のルールは、空のフィルターを指定することで、バケット内のすべてのオブジェクトを示しています。
- 2 番目のルールは、キー名プレフィックス (logs/) を指定することで、オブジェクトのサブセットのみを示しています。

ルール 1 では、すべてのオブジェクトを作成後 1 年で削除するように Amazon S3 にリクエストします。ルール 2 では、オブジェクトのサブセットを作成後 30 日で S3 Standard – IA ストレージクラスに移行するように Amazon S3 にリクエストします。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>30</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

この場合、競合は発生しないため、Amazon S3 は作成から 30 日後に logs/ プレフィックスを持つオブジェクトを S3 標準 – IA ストレージクラスへ移行させます。オブジェクトの作成後 1 年に達すると、そのオブジェクトは削除されます。

Example 2: 競合するライフサイクルアクション

この設定例には、オブジェクトの有効期間内の同じ時点で同じオブジェクトセットに対して 2 つの異なるアクションを実行するよう Amazon S3 に指示する 2 つのルールがあります。

- 両方のルールで同じキー名プレフィックスが指定されているので、どちらのルールも同じオブジェクトのセットに適用されます。
- どちらのルールでも、オブジェクトが作成されてから同じ 365 日後にルールを適用するように指定されています。
- 1 つのルールは、オブジェクトを S3 Standard – IA ストレージクラスに移行するように Amazon S3 に指示しています。もう 1 つのルールは、同じタイミングでオブジェクトを有効期限切れにするよう Amazon S3 に指示しています。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>365</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

この場合、ユーザーはオブジェクトを有効期限切れに (削除予定) するよう望んでおり、ストレージクラスを変更しても意味がないため、Amazon S3 はこれらのオブジェクトに対して失効アクションを選択します。

Example 3: ライフサイクルアクションが競合する重複したプレフィックス

この例の設定には、次のように重複するプレフィックスを指定する 2 つのルールがあります。

- ルール 1 では、空のプレフィックス (すべてのオブジェクトを示します) が指定されています。
- ルール 2 では、すべてのオブジェクトのサブセットを識別するキー名プレフィックス (logs/) が指定されています。

キー名プレフィックスが logs/ のオブジェクトのサブセットには、両方のルールの S3 ライフサイクルアクションが適用されます。片方のルールは作成されてから 10 日後にオブジェクトを移行するよう Amazon S3 に指示しており、もう一方のルールは作成されてから 365 日後にオブジェクトを移行するよう Amazon S3 に指示しています。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>10</Days>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>365</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

この場合、Amazon S3 は作成後 10 日での移行を選択します。

Example 4: タグベースのフィルタ処理の結果によるライフサイクルアクションの競合

それぞれがタグフィルターを指定する 2 つのルールを含む次のような S3 ライフサイクル設定があるものとします。

- ルール 1 では、タグベースのフィルタ (tag1/value1) が指定されています。このルールは、作成されてから 365 日後にオブジェクトを S3 Glacier Flexible Retrieval ストレージクラスに移行するよう Amazon S3 に指示します。
- ルール 2 では、タグベースのフィルタ (tag2/value2) が指定されています。このルールは、作成されてから 14 日後にオブジェクトを期限切れにするよう Amazon S3 に指示します。

S3 ライフサイクル設定は以下のようになります。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Tag>
         <Key>tag1</Key>
         <Value>value1</Value>
      </Tag>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>GLACIER<StorageClass>
      <Days>365</Days>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Tag>
         <Key>tag2</Key>
         <Value>value2</Value>
      </Tag>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>14</Days>
    </Expiration>
   </Rule>
```

```
</LifecycleConfiguration>
```

オブジェクトに両方のタグがある場合、Amazon S3 はどのルールに従うかを決定する必要があります。この場合、Amazon S3 は作成されてから 14 日後にオブジェクトを期限切れにします。オブジェクトは削除されるため、移行アクションは行われません。

> **Important**
>
> S3 ライフサイクル設定に複数のルールがある場合、1 つのオブジェクトが同じ日に複数の S3 ライフサイクルアクションの対象になることがあります。このような場合、Amazon S3 は以下の一般的なルールに従います。
>
> - 完全な削除は、移行より優先されます。
> - 移行は、削除マーカーの作成より優先されます。
> - オブジェクトが S3 Glacier Flexible Retrieval と S3 Standard-IA (または S3 One Zone-IA) 移行の両方の対象になる場合、Amazon S3 は S3 Glacier Flexible Retrieval 移行を選択します。
>
> 例については、「例 5: 重複するフィルター、競合するライフサイクルアクション、Amazon S3 がバージョン管理されていないバケットに対して行うこと」を参照してください。

例 6: バージョニングが有効なバケットへのライフサイクルルールを指定する

バージョニングが有効なバケットがあるとします。つまり、各オブジェクトに、最新のバージョンと以前のバージョンが 0 個以上あります。(S3 バージョニングの詳細については、S3 バケットでのバージョニングの使用 を参照してください。) この例では、1 年に相当する期間保持した後、以前のバージョンを削除するとします。S3 ライフサイクル設定では、任意のオブジェクトを 1 ~ 100 バージョンまで保持できます。

ストレージコストを節約するために、最新でないバージョンが最新でなくなってから 30 日後に、S3 Glacier Flexible Retrieval に移動する方法 (これらの最新でないオブジェクトは、リアルタイムアクセスが必要ないコールドデータを想定しています) 。また、最新のバージョンへのアクセス頻度は作成されてから 90 日後に低下すると予想されるため、これらのオブジェクトを S3 Standard-IA ストレージクラスに移動することもできます。

```
<LifecycleConfiguration>
```

```
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
           <Prefix></Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>90</Days>
           <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <NoncurrentVersionTransition>
            <NoncurrentDays>30</NoncurrentDays>
            <StorageClass>GLACIER</StorageClass>
        </NoncurrentVersionTransition>
       <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>5</NewerNoncurrentVersions>
            <NoncurrentDays>365</NoncurrentDays>
       </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

例 7: 期限切れオブジェクト削除マーカーを削除する

バージョニングが有効なバケットで、各オブジェクトに最新のバージョン 1 個と、最新でないバージョンが 0 個以上存在します。オブジェクトを削除するときは、次のことに注意してください。

- 削除リクエストでバージョン ID を指定しない場合、Amazon S3 はオブジェクトを削除する代わりに削除マーカーを追加します。現在のオブジェクトバージョンが最新でなくなり、削除マーカーが現在のバージョンになります。
- 削除リクエストでバージョン ID を指定した場合、Amazon S3 はそのオブジェクトバージョンを完全に削除します (削除マーカーは作成されません)。
- 最新でないバージョンが 0 の削除マーカーは、有効期限切れオブジェクト削除マーカー と呼ばれます。

この例では、バケットに期限切れオブジェクト削除マーカーを作成できるシナリオと、S3 ライフサイクル設定を使用して期限切れオブジェクト削除マーカーを削除するよう Amazon S3 に指示する方法を示します。

以下に示すように、最新でないバージョンがそのようになってから 30 日後に削除する NoncurrentVersionExpiration アクションを使用し、最大 10 個を保持する S3 ライフサイクル設定を記述するとします。

```
<LifecycleConfiguration>
    <Rule>
        ...
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

NoncurrentVersionExpiration アクションは、現在のオブジェクトバージョンには適用されません。最新以外のバージョンのみ削除されます。

現在のオブジェクトバージョンの場合、現在のオブジェクトバージョンが明確に定義されたライフサイクルに従うかどうかに応じて存続期間を管理する以下のオプションがあります。

- 現在のオブジェクトバージョンが明確に定義されたライフサイクルに従う。

  この場合、次の例に示すように、現在のバージョンを削除するよう Amazon S3 に指示する Expiration アクションを含む S3 ライフサイクル設定を使用できます。

  ```
  <LifecycleConfiguration>
      <Rule>
          ...
          <Expiration>
             <Days>60</Days>
          </Expiration>
          <NoncurrentVersionExpiration>
              <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
              <NoncurrentDays>30</NoncurrentDays>
          </NoncurrentVersionExpiration>
      </Rule>
  </LifecycleConfiguration>
  ```

  この例では、Amazon S3 は、現在の各オブジェクトバージョンに削除マーカーを追加することで、作成から 60 日後に現在のバージョンを削除します。このプロセスにより、現在のバージョン

が最新でなくなり、削除マーカーが現在のバージョンになります。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

> **Note**
>
> 同じルールに Days と ExpiredObjectDeleteMarker のタグの両方を指定することはできません。Amazon S3は削除マーカーが経過機関の基準を満たすと、Days タグを指定することで、自動的に ExpiredObjectDeleteMarker クリーンアップを実行します。削除マーカーが唯一のバージョンになったらすぐにクリーンアップできるよう、ExpiredObjectDeleteMarker タグだけを含む別のルールを作成します。

同じ S3 ライフサイクル設定の NoncurrentVersionExpiration アクションは、最新でないオブジェクトが最新でなくなってから 30 日後に削除します。したがって、この例では、すべてのオブジェクトバージョンがオブジェクトの作成から 90 日後に完全に削除されます。この過程で期限切れのオブジェクト削除マーカーが作成されますが、Amazon S3 が検知して削除します。

- 現在のオブジェクトバージョンに明確に定義されたライフサイクルがない。

  この場合、必要でない場合はオブジェクトを削除できます。このとき、1 つ以上の最新でないバージョンとともに削除マーカーが作成されます。NoncurrentVersionExpiration アクションを含む S3 ライフサイクル設定が最新でないすべてのバージョンを削除した場合、期限切れオブジェクト削除マーカーが生成されます。

  特にこのシナリオのために、S3 ライフサイクル設定には、期限切れオブジェクト削除マーカーを削除するために使用できる Expiration アクションを提供します。

```
<LifecycleConfiguration>
    <Rule>
       <ID>Rule 1</ID>
        <Filter>
          <Prefix>logs/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Expiration>
           <ExpiredObjectDeleteMarker>true</ExpiredObjectDeleteMarker>
        </Expiration>
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
```

```
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

`ExpiredObjectDeleteMarker` アクションで `true` 要素を `Expiration` に設定することで、期限切れオブジェクト削除マーカーを削除するよう Amazon S3 に指示できます。

> **Note**
>
> `ExpiredObjectDeleteMarker` S3 ライフサイクルアクションを指定するときは、ルールではタグベースのフィルターを指定できません。

例 8: マルチパートアップロードを中止するライフサイクル設定

Amazon S3 マルチパートアップロード REST API オペレーションを使用すると、大容量オブジェクトをいくつかに分けてアップロードできます。マルチパートアップロードの詳細については、「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。

S3 ライフサイクル設定を使用すると、不完全なマルチパートアップロード (ルールで指定したキー名プレフィックスで特定) が開始後指定した日数以内に完了しない場合、アップロードを停止するように Amazon S3 に指示できます。Amazon S3 は、マルチパートアップロードを中止するとき、マルチパートアップロードに関連付けられているすべてのパートを削除します。このプロセスでは、Amazon S3 に保存されているパートを含む不完全なマルチパートアップロードがないことが保証されるので、ストレージコストを制御するのに役立ちます。

> **Note**
>
> `AbortIncompleteMultipartUpload` S3 ライフサイクルアクションを指定するときは、ルールではタグベースのフィルターを指定できません。

`AbortIncompleteMultipartUpload` アクションにルールを指定する S3 ライフサイクル設定の例を次に示します。このアクションは、開始されたから 7 日後に不完全なマルチパートアップロードを停止するように Amazon S3 に指示します。

```
<LifecycleConfiguration>
    <Rule>
```

```
        <ID>sample-rule</ID>
        <Filter>
           <Prefix>SomeKeyPrefix/</Prefix>
        </Filter>
        <Status>rule-status</Status>
        <AbortIncompleteMultipartUpload>
          <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

例 9: サイズベースのルールを使用したライフサイクル設定

オブジェクトのサイズのみに基づいてオブジェクトを移行するルールを作成できます。オブジェクトのサイズは、最小 (ObjectSizeGreaterThan) または最大 (ObjectSizeLessThan) を指定することも、バイト数で範囲を指定することもできます。プレフィックスとサイズの規則のように、複数のフィルターを使用する場合は、フィルターを <And> 要素で囲む必要があります。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Transition with a prefix and based on size</ID>
    <Filter>
       <And>
          <Prefix>tax/</Prefix>
          <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
       </And>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

ObjectSizeGreaterThan と ObjectSizeLessThan の両方の要素を使用して範囲を指定する場合、最大オブジェクトサイズは最小オブジェクトサイズより大きくなければなりません。複数のフィルターを使用する場合は、フィルターを <And> 要素で囲む必要があります。次の例は、500 バイトから 64,000 バイトの範囲でオブジェクトを指定する方法を示しています。範囲を指定する場合、ObjectSizeGreaterThan フィルターと ObjectSizeLessThan フィルターは指定された値を除外します。詳細については、「the section called “Filter 要素”」を参照してください。

```
<LifecycleConfiguration>
    <Rule>
        ...
          <And>
             <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
             <ObjectSizeLessThan>64000</ObjectSizeLessThan>
          </And>
    </Rule>
</LifecycleConfiguration>
```

バージョニングが有効になっているバケットで作成された最新でない削除マーカーオブジェクトなど、データがなく、最新でないオブジェクトに限り期限切れにするルールを作成することもできます。NoncurrentVersionExpression アクションを使用して、最新でないバージョンが最新のバージョンでなくなってから 30 日後に削除し、オブジェクトの最新でないバージョンの保持を最大 10 個にする例は、次のとおりです。また、ObjectSizeLessThan エレメントを使用してデータのないオブジェクトのみをフィルタリングします。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Expire noncurrent with size less than 1 byte</ID>
    <Filter>
       <ObjectSizeLessThan>1</ObjectSizeLessThan>
    </Filter>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
       <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
       <NoncurrentDays>30</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

Amazon S3 インベントリ

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されま

す。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「Default encryption FAQ」(デフォルト暗号化に関するよくある質問) を参照してください。

Amazon S3 インベントリを使用してストレージを管理できます。例えば、ビジネス、コンプライアンス、および規制上のニーズに対応して、オブジェクトのレプリケーションや暗号化のステータスを監査およびレポートするために使用できます。また、Amazon S3 同期 `List` API オペレーションのスケジュールされた代替手段として Amazon S3 インベントリを使用し、ビジネスワークフローやビッグデータジョブを簡素化、高速化できます。Amazon S3 インベントリは、`List` API オペレーションを使用してオブジェクトを監査しないため、バケットのリクエストレートには影響しません。

Amazon S3 インベントリは、カンマ区切り値 (CSV)、Apache Optimized Row Columnar (ORC)、または Apache Parquet 出力ファイルを通じて、S3 バケットや共有プレフィックス (オブジェクト名の先頭が共通文字列) を持つオブジェクトについて、オブジェクトおよび対応するメタデータを毎日または毎週一覧表示します。毎週のインベントリを設定すると、最初のレポートの後は毎週日曜日 (UTC タイムゾーン) にレポートが生成されます。Amazon S3 インベントリの料金に関する詳細については、Amazon S3 の料金 を参照してください。

1 つのバケットに対して複数のインベントリリストを設定できます。インベントリリストを設定する場合、以下を指定できます。

- インベントリに含めるオブジェクトメタデータ
- すべてのオブジェクトバージョンを一覧表示するか、現在のバージョンのみを一覧表示するか
- インベントリリストファイル出力を保存する場所
- インベントリを毎日生成するか、毎週生成するか
- インベントリリストファイルを暗号化するかどうか

Amazon S3 インベントリを標準 SQL クエリを使用してクエリするには、Amazon Athena、Amazon Redshift Spectrum、その他のツール (Presto、Apache Hive、Apache Spark など) を使用できます。Athena を使用してインベントリファイルをクエリする方法の詳細については、「the section called “Athena でインベントリをクエリする”」を参照してください。

ソースバケットと保存先バケット

インベントリでオブジェクトをリストする対象のバケットは、ソースバケットと呼ばれます。インベントリリストファイルを保存する先のバケットは、保存先バケットと呼ばれます。

ソースバケット

インベントリは、ソースバケットに格納されているオブジェクトをリストします。バケット全体のインベントリリストを取得することも、オブジェクトキー名のプレフィックスでリストをフィルタリングすることもできます。

ソースバケット:

- インベントリにリストされているオブジェクトが含まれます。
- インベントリの設定が含まれます。

保存先バケット

Amazon S3 インベントリリストのファイルは、保存先バケットに書き込まれます。すべてのインベントリリストファイルを保存先バケットの共通の場所にグループ化するには、インベントリ設定で保存先のプレフィックスを指定できます。

保存先バケット:

- インベントリのファイルリストが含まれます。
- 保存先バケットに保存されているすべてのインベントリリストファイルを一覧表示するマニフェストファイルが含まれます。詳細については、「インベントリマニフェスト」を参照してください。
- バケットの所有権を検証するためのアクセス許可と、バケットにファイルを書き込むためのアクセス許可を Amazon S3 に付与する、バケットポリシーが必要です。
- ソースバケットと同じ AWS リージョン に存在する必要があります。
- ソースバケットと同じでもかまいません。
- ソースバケットを所有するアカウントとは別の AWS アカウント によって所有されていてもかまいません。

Amazon S3 インベントリのリスト

インベントリリストファイルには、ソースバケット内のオブジェクトのリストと、各オブジェクトのメタデータが含まれます。インベントリリストファイルは、次のいずれかの形式で保存先バケットに保存されます。

- GZIP で圧縮された CSV ファイル
- ZLIB で圧縮された Apache Optimized Row Columnar (ORC)
- Snappy で圧縮された Apache Parquet ファイル

> **Note**
>
> Amazon S3 インベントリレポートのオブジェクトのソート結果は一切保証されません。

インベントリリストファイルには、ソースバケット内のオブジェクトのリストと、リストされた各オブジェクトのメタデータが含まれます。

- バケット名 – インベントリ対象のバケットの名前。
- キー名 – バケット内のオブジェクトを一意に識別するオブジェクトのキー名 (またはキー)。CSV ファイル形式を使用すると、キー名は URL エンコードされるため、これをデコードしてから使用する必要があります。
- バージョン ID – オブジェクトのバージョン ID。バケットのバージョニングを有効にすると、Amazon S3 はバケットに追加されたオブジェクトにバージョン番号を割り当てます。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。リストを現在のバージョンのオブジェクトのみに設定した場合、このフィールドは含まれません。
- IsLatest – オブジェクトが現在のバージョンのオブジェクトである場合は、`True` に設定されます リストを現在のバージョンのオブジェクトのみに設定した場合、このフィールドは含まれません。
- 削除マーカー – オブジェクトが削除マーカーである場合は、`True` に設定されます。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。(オブジェクトのすべてのバージョンを含めるようにレポートを設定している場合、このフィールドはレポートに自動的に追加されます)。
- サイズ - バイト単位のオブジェクトサイズ。不完全なマルチパートアップロード、オブジェクトメタデータ、削除マーカーのサイズは含まれません。
- 最終更新日 – オブジェクトの作成日または最終更新日のどちらか新しい方。

- ETag – エンティティタグ (ETag) は、オブジェクトのハッシュです。ETag は、変更をオブジェクトのコンテンツにのみ反映し、メタデータには反映しません。ETag は、オブジェクトデータの MD5 ダイジェストである場合があります。どちらであるかは、オブジェクトの作成方法と暗号化方法によって決まります。
- ストレージクラス – オブジェクトの保存に使用するストレージクラス。STANDARD、REDUCED_REDUNDANCY、STANDARD_IA、ONEZONE_IA、INTELLIGENT_TIERING、GLACIER または SNOW に設定します。詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。
- マルチパートアップロードフラグ – オブジェクトがマルチパートアップロードとしてアップロードされた場合は、True に設定されます。詳細については、「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。
- レプリケーションステータス — PENDING、COMPLETED、FAILED または REPLICA に設定します。詳細については、「レプリケーションステータス情報の取得」を参照してください。
- 暗号化ステータス — サーバー側の暗号化のステータス。使用する暗号化キーの種類が Amazon S3 マネージド (SSE-S3) キー、AWS Key Management Service (AWS KMS) キー (SSE-KMS)、カスタマー提供キー (SSE-C) のいずれであるかに応じて異なります。SSE-S3、SSE-C、SSE-KMS、または NOT-SSE に設定します。ステータスが NOT-SSE の場合、オブジェクトはサーバー側の暗号化を使用して暗号化されません。詳細については、「暗号化によるデータの保護」を参照してください。
- S3 オブジェクトロック: 日付までの保持期限 – ロックされたオブジェクトを削除できなくなる日付。詳細については、「S3 オブジェクトロックの使用」を参照してください。
- S3 オブジェクトロック: 保持モード – ロックされたオブジェクトの Governance または Compliance に設定します。詳細については、「S3 オブジェクトロックの使用」を参照してください。
- S3 オブジェクトロック: リーガルホールドステータス – リーガルホールドがオブジェクトに適用されている場合は On に設定します。それ以外の場合は、Off に設定されます。詳細については、「S3 オブジェクトロックの使用」を参照してください。
- S3 Intelligent-Tiering: アクセス階層 – S3 Intelligent-Tiering ストレージクラスがオブジェクトの保存先である場合、オブジェクトのアクセス階層 (高頻度または低頻度)。FREQUENT、INFREQUENT、ARCHIVE_INSTANT_ACCESS、ARCHIVE、または DEEP_ARCHIVE に設定します。詳細については、「アクセスパターンが変化する、またはアクセスパターンが不明なデータを、自動的に最適化するためのストレージクラス」を参照してください。

- S3 バケットキーステータス — ENABLED または DISABLED に設定します。オブジェクトが SSE-KMS に S3 バケットキーを使用するかどうかを示します。詳細については、「Amazon S3 バケットキーの使用」を参照してください。
- チェックサムアルゴリズム - オブジェクトのチェックサムを作成するために使用されるアルゴリズムを示します。
- オブジェクトアクセスコントロールリスト — このオブジェクトへのアクセスを許可する AWS アカウントまたはグループと、許可するアクセスの種類を定義するアクセスコントロールリスト (ACL)。オブジェクト ACL フィールドは JSON 形式で定義します。S3 インベントリレポートには、ACL がバケットで無効になっている場合でも、ソースバケット内のオブジェクトに関連付けられた ACL が含まれます。詳細については、オブジェクト ACL フィールドの使用およびアクセスコントロールリスト (ACL) の概要を参照してください。

> **Note**
>
> オブジェクト ACL フィールドは JSON 形式で定義します。インベントリレポートには、オブジェクト ACL フィールドの値が base64 エンコードされた文字列として表示されます。
> 例えば、次のオブジェクト ACL フィールドが JSON 形式で存在するとします。
>
> ```
> {
>         "version": "2022-11-10",
>         "status": "AVAILABLE",
>         "grants": [{
>             "canonicalId": "example-canonical-user-ID",
>             "type": "CanonicalUser",
>             "permission": "READ"
>         }]
> }
> ```
>
> オブジェクト ACL フィールドはエンコードされ、次の base64 エンコードされた文字列として表示されます。
>
> ```
> eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3siY2Fub25pY2Fs
> ```
>
> オブジェクト ACL フィールドのデコードされた値を JSON で取得するには、Amazon Athena でこのフィールドをクエリできます。クエリの例については、「Amazon Athena で Amazon S3 インベントリをクエリする」を参照してください。

- オブジェクト所有者 — オブジェクトの所有者。

> **Note**
>
> ライフサイクル設定に基づいて、オブジェクトの存続期間が終了すると、Amazon S3 が削除キューに追加し、非同期的に削除します。そのため、有効期限が切れる日と Amazon S3 がオブジェクトを削除する日との間に遅延が生じることがあります。インベントリレポートには、有効期限が切れていてもまだ削除されていないオブジェクトが含まれます。S3 ライフサイクルの有効期限アクションの詳細については、「オブジェクトの有効期限」を参照してください。

古いインベントリリストを削除するライフサイクルポリシーを作成することをお勧めします。詳細については、「ストレージのライフサイクルの管理」を参照してください。

`s3:PutInventoryConfiguration` アクセス許可により、ユーザーは、インベントリリストを設定するとき、各オブジェクトについて以前にリストされたすべてのメタデータフィールドを選択することと、インベントリを保存する保存先バケットを指定することの両方ができるようになります。保存先バケット内のオブジェクトへの読み取りアクセス権を持つユーザーは、インベントリリストで利用可能なすべてのオブジェクトメタデータフィールドにアクセスできます。インベントリレポートへのアクセスを制限するには、「S3 インベントリおよび S3 分析に対するアクセス許可の付与」を参照してください。

インベントリ整合性

すべてのオブジェクトが各インベントリリストに表示されない場合があります。インベントリリストは、PUT リクエスト (新しいオブジェクトと上書きの両方) および DELETE リクエストの結果整合性を提供します。バケットの各インベントリリストは、バケット項目のスナップショットです。これらのリストは結果的に整合します (つまり、リストには最近追加または削除されたオブジェクトが含まれない場合があります)。

オブジェクトに対してアクションを実行する前に、オブジェクトの状態を検証するには、`HeadObject` REST API リクエストを実行してオブジェクトのメタデータを取得するか、Amazon S3 コンソールでオブジェクトのプロパティを確認することをお勧めします。AWS CLI または AWS SDK でオブジェクトのメタデータを確認することもできます。詳細については、「Amazon Simple Storage Service API リファレンス」の「`HeadObject`」を参照してください。

Amazon S3インベントリの使用のさらなる詳細については、以下のトピックを参照してください。

トピック

Amazon S3 インベントリの設定

Amazon S3 インベントリには、定義したスケジュールに従ってオブジェクトとメタデータのフラットファイルリストが用意されています。Amazon S3 同期 `List` API オペレーションの代わりにスケジュールされた S3 インベントリを使用できます。S3 インベントリは、カンマ区切り値 (CSV)、Apache Optimized Row Columnar (ORC)、または Apache Parquet (Parquet) 出力ファイルを使用して、オブジェクトおよび対応するメタデータを一覧表示します。

S3 インベントリは、S3 バケットまたはプレフィックスを共有するオブジェクト (つまり、名前の先頭が同じ文字列のオブジェクト) に関して、インベントリリストが用意されています。詳細については、「Amazon S3 インベントリ」を参照してください。

このセクションでは、インベントリのソースバケットと保存先バケットに関する詳細を含め、インベントリを設定する方法について説明します。

トピック

概要

Amazon S3 インベントリは、定義したスケジュールで S3 バケット内のオブジェクトのリストを作成し、ストレージの管理に役立ちます。1 つのバケットに対して複数のインベントリリストを設定で

きます。インベントリリストは、保存先バケットの CSV、ORC、または Parquet ファイルに発行されます。

インベントリを設定する最も簡単な方法は、Amazon S3 を使用することですが、REST API、AWS Command Line Interface (AWS CLI)、または AWS SDK を使用することもできます。コンソールは、次の手順の最初のステップを実行します。保存先バケットへのバケットポリシーの追加。

S3 バケットの Amazon S3 インベントリをセットアップする方法

1. 保存先バケットのバケットポリシーを追加します。

   定義された場所のバケットにオブジェクトを書き込むアクセス許可を Amazon S3 に付与するバケットポリシーを保存先バケットに作成する必要があります。ポリシーの例については、「S3 インベントリおよび S3 分析に対するアクセス許可の付与」を参照してください。
2. ソースバケットのオブジェクトをリストして保存先バケットにリストを発行するようにインベントリを設定します。

   ソースバケットのインベントリリストを設定するときは、リストを保存する保存先バケットと、リストを毎日または毎週のどちらで生成したいかを、指定します。また、すべてのオブジェクトバージョンまたは現在のバージョンだけのどちらをリストするか、どのオブジェクトメタデータを含めるかも設定できます。

   S3 インベントリレポート設定の一部のオブジェクトメタデータフィールドはオプションです。つまり、デフォルトで使用できますが、ユーザーに `s3:PutInventoryConfiguration` アクセス許可を付与すると制限できます。`s3:InventoryAccessibleOptionalFields` 条件キーを使用して、ユーザーがこれらのオプションのメタデータフィールドをレポートに含めることができるかどうかを制御できます。

   S3 インベントリで使用できるオプションのメタデータフィールドの詳細については、「Amazon Simple Storage Service API リファレンス」の「OptionalFields」を参照してください。インベントリ設定の特定のオプションのメタデータフィールドへのアクセスを制限する方法についての詳細は、「S3 インベントリレポート設定の作成を制御する」を参照してください。

   Amazon S3 マネージド キー (SSE-S3) または AWS Key Management Service (AWS KMS) カスタマーマネージドキー (SSE-KMS) を使用したサーバー側の暗号化により、インベントリリストファイルを暗号化するように指定できます。

> **Note**
>
> AWS マネージドキー(aws/s3) は S3 インベントリでの SSE-KMS 暗号化ではサポートされていません。

SSE-S3 および SSE-KMS に関する詳細は、「サーバー側の暗号化によるデータの保護」を参照してください。SSE-KMS 暗号化を使用する場合は、ステップ 3 を参照してください。

- コンソールを使用してインベントリリストを設定する方法については、「S3 コンソールを使用したインベントリの設定」を参照してください。
- Amazon S3 API を使用してインベントリリストを設定するには、PutBucketInventoryConfiguration REST API オペレーションを使用するか、AWS CLI または AWS SDK の同等のオペレーションを使用します。

3. SSE-KMS を使用してインベントリリストファイルを暗号化するには、AWS KMS key を使用するための許可を Amazon S3 に付与します。

   Amazon S3 REST API、AWS CLI、または AWS SDK を使用してインベントリリストファイルの暗号化を設定することができます。どちらの方法を選択しても、 カスタマーマネージドキーを使用してインベントリファイルを暗号化する許可を Amazon S3 に付与する必要があります。Amazon S3 に許可を付与するには、インベントリファイルの暗号化に使用するカスタマーマネージドキーのキーポリシーを変更します。詳細については、「カスタマーマネージドキーを暗号化に使用するためのアクセス許可を Amazon S3 に付与する」を参照してください。

   インベントリリストファイルを保存する先のバケットは、ソースバケットを所有するアカウントとは異なる AWS アカウントが所有している場合があります。Amazon S3 インベントリのクロスアカウントオペレーションで SSE-KMS 暗号化を使用する場合は、S3 インベントリの設定時に、完全修飾された KMS キー ARN を使用することをお勧めします。詳細については、「Amazon Simple Storage Service API リファレンス」の「クロスアカウント操作での SSE-KMS 暗号化の使用」と「ServerSideEncryptionByDefault」を参照してください。

ターゲットバケットポリシーの作成

Amazon S3 コンソールを使用してインベントリ設定を作成する場合、Amazon S3 は、バケットへの書き込みアクセス許可を Amazon S3 に付与するバケットポリシーを、保存先バケットに自動的に作成します。ただし、AWS CLI、AWS SDK、または Amazon S3 REST API を使用してインベントリ設定を作成する場合は、バケットポリシーを保存先バケットに手動で追加する必要があります。詳

細については、「S3 インベントリおよび S3 分析に対するアクセス許可の付与」を参照してください。S3 インベントリの送信先バケットポリシーにより、Amazon S3 がインベントリレポートのデータをバケットに書き込むことができるようになります。

バケットポリシーを作成しようとしたときにエラーが発生した場合、解決するための手順が表示されます。例えば、別の AWS アカウント 内のターゲットバケットを選択したが、バケットポリシーの読み書きを行う許可がない場合、次のエラーメッセージが表示されます。

この場合、ターゲットバケットの所有者は、バケットポリシーをターゲットバケットに追加する必要があります。ポリシーがターゲットバケットに追加されない場合、Amazon S3 にはターゲットバケットに書き込むアクセス許可がないため、インベントリレポートを取得できません。ソースバケットが現在のユーザー以外のアカウントによって所有されている場合、ポリシーでソースバケット所有者の正しいアカウント ID に置き換える必要があります。

カスタマーマネージドキーを暗号化に使用するためのアクセス許可を Amazon S3 に付与する

AWS Key Management Service (AWS KMS) カスタマーマネージドキーを使用して暗号化する許可を Amazon S3 に付与するには、キーポリシーを使用する必要があります。カスタマーマネージドキーを使用できるようにキーポリシーを更新するには、次の手順に従います。

カスタマーマネージドキーを使用して暗号化するアクセス許可を Amazon S3 に付与するには

1. カスタマーマネージドキーを所有する AWS アカウント を使用して、AWS Management Console にサインインします。
2. AWS KMS コンソール (https://console.aws.amazon.com/kms) を開きます。
3. AWS リージョン を変更するには、ページの右上隅にあるリージョンセレクターを使用します。
4. 左のナビゲーションペインで、[Customer managed keys] を選択します。
5. [カスタマーマネージドキー] で、インベントリファイルの暗号化に使用するカスタマーマネージドキーを選択します。
6. [Key Policy] (キーポリシー) セクションで、[Switch to policy view] (ポリシービューへの切り替え) を選択します。
7. [編集] をクリックし、キーポリシーを更新します。
8. [キーポリシーの編集] ページで、既存のキーポリシーに以下の行を追加します。*source-account-id* および *example-s3-source-bucket* として、ユースケースに応じた値を入力します。

```
{
    "Sid": "Allow Amazon S3 use of the customer managed key",
    "Effect": "Allow",
    "Principal": {
        "Service": "s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition":{
      "StringEquals":{
         "aws:SourceAccount":"source-account-id"
     },
      "ArnLike":{
        "aws:SourceARN": "arn:aws:s3:::example-s3-source-bucket"
     }
   }
}
```

9. [Save changes] (変更の保存) をクリックします。

カスタマーマネージドキーの作成とキーポリシーの使用の詳細については、AWS Key Management Service デベロッパーガイドの次のリンクを参照してください。

- キーの管理
- AWS KMS のキーポリシー

S3 コンソールを使用したインベントリの設定

S3 コンソールを使用してインベントリを設定するには、次の手順を使用します。

> **Note**
>
> Amazon S3 が最初のインベントリレポートを配信するまでに最大 48 時間かかることがあります。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。[バケット] リストで、Amazon S3 インベントリを設定する対象のバケットの名前を選択します。
3. [管理] タブを選択します。
4. [インベントリ設定] で、[インベントリ設定の作成] を選択します。
5. [インベントリ設定名] に、名前を入力します。
6. [インベントリスコープ] で、次の操作を行います。

   - オプションのプレフィックスを入力します。
   - 含めるオブジェクトバージョンとして、[現在のバージョンのみ] または [すべてのバージョンを含める] のどちらかを選択します。
7. [Report details] (レポートの詳細) で、[This account] (このアカウント) または [A different account] (別のアカウント) のいずれかを、レポートを保存する AWS アカウント の場所として選択します。
8. [保存先] で、インベントリレポートを保存する先のバケットを選択します。

   ターゲットバケットは、インベントリをセットアップするバケットと同じ AWS リージョン にある必要があります。ターゲットバケットは、別の AWS アカウント にある場合があります。保存先バケットを指定するときに、インベントリレポートをグループ化するためのオプションのプレフィックスを含めることもできます。

   [送信先] バケットフィールドの下に、[送信先バケットのアクセス許可] と表示されます。このアクセス許可を送信先バケットポリシーに追加し、Amazon S3 がバケットにデータを配置できるようにします。詳細については、「ターゲットバケットポリシーの作成」を参照してください。
9. [頻度] の下で、レポートを生成する頻度として、[毎日] または [毎週] を選択します。
10. [出力形式] で、レポートの形式を以下から 1 つ選択します。

    - CSV — このインベントリレポートを S3 バッチオペレーションで使用するか、このレポートを Microsoft Excel などの別のツールで分析する場合は、[CSV] を選択します。
    - Apache ORC
    - Apache Parquet
11. [ステータス] の下で、[有効化] または [無効化] を選択します。
12. サーバー側の暗号化を設定するには、[インベントリ レポートの暗号化] で次の手順に従います。

a. [サーバー側の暗号化] で、[暗号化キーを指定しない] または [暗号化キーを指定する] を選択してデータを暗号化します。

- Amazon S3 にオブジェクトを格納するときに、バケット設定をデフォルトのサーバー側暗号化のままにするには、[暗号化キーを指定しない] を選択します。送信先バケットで S3 バケットキーが有効になっている限り、コピーオペレーションは送信先バケットに S3 バケットキーを適用します。

> **Note**
>
> 指定された宛先のバケットポリシーで、Amazon S3 に保存する前にオブジェクトを暗号化する必要がある場合は、[暗号化キーを指定する] を選択する必要があります。そうしないと、宛先へのオブジェクトのコピーが失敗します。

- オブジェクトを Amazon S3 に保存する前に暗号化するには、[暗号化キーを指定する] を選択します。

b. [暗号化キーを指定する] を選択した場合は、[暗号化タイプ] で [Amazon S3 マネージドキー (SSE-S3)] または [AWS Key Management Service キー (SSE-KMS)] を選択する必要があります。

SSE-S3 は、最強のブロック暗号の 1 つである 256 ビットの 高度暗号化規格 (AES-256) を使用して、各オブジェクトを暗号化します。SSE-KMS を使用すると、キーをより細かく制御できます。SSE-KMS に関する詳細は、「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)」を参照してください。SSE-KMS に関する詳細は、「AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用」を参照してください。

> **Note**
>
> SSE-KMS を使用してインベントリリストファイルを暗号化するには、カスタマーマネージドキーを使用するためのアクセス許可を Amazon S3 に付与する必要があります。手順については、「KMS キーを使用した暗号化のための Amazon S3 への許可の付与」を参照してください。

c. [AWS Key Management Service キー (SSE-KMS)] を選択した場合は、AWS KMS key で、以下のオプションのいずれかを使用して AWS KMS キーを指定できます。

> **Note**
>
> インベントリリストファイルを保存する送信先バケットが別の AWS アカウント によって所有されている場合は、完全修飾 KMS キー ARN を使用して KMS キーを指定するようにしてください。

- 使用可能な KMS キーのリストから選択するには、[AWS KMS キーから選択する] を選択し、使用可能なキーのリストから対称暗号化 KMS キーを選択します。KMS キーがバケットと同じリージョンにあることを確認します。

> **Note**
>
> AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がリストに表示されます。ただし、AWS マネージドキー (aws/s3) は S3 インベントリでの SSE-KMS 暗号化には対応していません。

- KMS キー ARN を入力するには、[AWS KMS キー ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。
- AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

13. [追加のメタデータフィールド] で、インベントリレポートに追加するフィールドを以下から 1 つ以上選択します。

    - サイズ - バイト単位のオブジェクトサイズ。不完全なマルチパートアップロード、オブジェクトメタデータ、削除マーカーのサイズは含まれません。
    - 最終更新日 – オブジェクトの作成日または最終更新日のどちらか新しい方。
    - [マルチパートアップロード] – オブジェクトがマルチパートアップロードとしてアップロードされたことを指定します。詳細については、「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。
    - [レプリケーションステータス] – オブジェクトのレプリケーションステータス。詳細については、「レプリケーションステータス情報の取得」を参照してください。
    - 暗号化ステータス – オブジェクトの暗号化に使用するサーバー側の暗号化タイプ。詳細については、「サーバー側の暗号化によるデータの保護」を参照してください。

- バケットキーのステータス — AWS KMS で生成したバケットレベルのキーをオブジェクトに適用するかどうかを示します。詳細については、「Amazon S3 バケットキーを使用した SSE-KMS のコストの削減」を参照してください。
- オブジェクトアクセスコントロールリスト — このオブジェクトへのアクセスを許可する AWS アカウントまたはグループと、許可するアクセスのタイプを定義するオブジェクトごとのアクセスコントロールリスト (ACL)。このフィールドの詳細については、「オブジェクト ACL フィールドの使用」を参照してください。ACL の詳細については、「アクセスコントロールリスト (ACL) の概要」を参照してください。
- オブジェクト所有者 — オブジェクトの所有者。
- ストレージクラス – オブジェクトの保存に使用するストレージクラス。
- インテリジェントな階層化: アクセス階層 – S3 Intelligent-Tiering ストレージクラスがオブジェクトの保存先である場合、オブジェクトのアクセス階層 (高頻度または低頻度) を示します。詳細については、「アクセスパターンが変化する、またはアクセスパターンが不明なデータを、自動的に最適化するためのストレージクラス」を参照してください。
- ETag – エンティティタグ (ETag) は、オブジェクトのハッシュです。ETag は、変更をオブジェクトのコンテンツにのみ反映し、そのメタデータには反映しません。ETag は、オブジェクトデータの MD5 ダイジェストである場合と、そうでない場合があります。どちらであるかは、オブジェクトの作成方法と暗号化方法によって決まります。詳細については、「Amazon Simple Storage Service API リファレンス」の「Object」を参照してください。
- チェックサムアルゴリズム — オブジェクトのチェックサムを作成するために使用されるアルゴリズムを示します。
- [すべてのオブジェクトロック設定] – オブジェクトのオブジェクトロックステータス (以下の設定が含まれます)。
  - オブジェクトロック: 保持モード – オブジェクトに適用される保護のレベル ([ガバナンス] または [コンプライアンス])。
  - オブジェクトロック: 日付までの保持期限 – ロックされたオブジェクトを削除できなくなる日付。
  - [オブジェクトロック: リーガルホールドステータス] – ロックされたオブジェクトのリーガルホールドステータス。

  S3 オブジェクトロックの詳細については、「S3 オブジェクトロックの仕組み」を参照してください。

インベントリレポートの内容の詳細については、「Amazon S3 インベントリのリスト」を参照してください。

インベントリ設定の特定のオプションのメタデータフィールドへのアクセスを制限する方法についての詳細は、「S3 インベントリレポート設定の作成を制御する」を参照してください。

14. [Create] (作成) を選択します。

インベントリリストが発行されると、Amazon S3 Select を使用してインベントリリストファイルにクエリを実行できます。Amazon S3 Select を使用してインベントリリストを見つけ、インベントリリストファイルにクエリを実行する方法の詳細については、「インベントリリストの検索」を参照してください。

REST API を使用して S3 インベントリを操作する

次に示すのは、Amazon S3 インベントリの操作に使用できる REST オペレーションです。

- DeleteBucketInventoryConfiguration
- GetBucketInventoryConfiguration
- ListBucketInventoryConfigurations
- PutBucketInventoryConfiguration

インベントリ完了に関する Amazon S3 イベント通知の設定

マニフェストチェックサムファイルが作成されたら通知を受け取るように、Amazon S3 イベント通知をセットアップできます。これは、インベントリリストが保存先バケットに追加されたことを示します。マニフェストは、保存先の場所にあるすべてのインベントリリストの最新のリストです。

Amazon S3 は、Amazon Simple Notification Service (Amazon SNS) トピック、Amazon Simple Queue Service (Amazon SQS) キュー、または AWS Lambda 関数にイベントを発行できます。詳細については、「Amazon S3 イベント通知」を参照してください。

次の通知設定は、保存先バケットに新しく追加されるすべての `manifest.checksum` ファイルが AWS Lambda `cloud-function-list-write` によって処理されることを定義します。

```
<NotificationConfiguration>
  <QueueConfiguration>
```

```
      <Id>1</Id>
      <Filter>
          <S3Key>
              <FilterRule>
                  <Name>prefix</Name>
                  <Value>destination-prefix/source-bucket</Value>
              </FilterRule>
              <FilterRule>
                  <Name>suffix</Name>
                  <Value>checksum</Value>
              </FilterRule>
          </S3Key>
      </Filter>
      <Cloudcode>arn:aws:lambda:us-west-2:222233334444:cloud-function-list-write</
Cloudcode>
      <Event>s3:ObjectCreated:*</Event>
  </QueueConfiguration>
  </NotificationConfiguration>
```

詳細については、AWS Lambda デベロッパーガイドの Amazon S3 で AWS Lambda を使用するを参照してください。

インベントリリストの検索

インベントリリストが発行されると、マニフェストファイルは保存先バケットの次の場所に発行されます。

```
destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.json
 destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.checksum
 destination-prefix/source-bucket/config-ID/hive/dt=YYYY-MM-DD-HH-MM/symlink.txt
```

- *destination-prefix* は、インベントリ設定でオプションで指定するオブジェクトキー名プレフィックスです。このプレフィックスを使用して、保存先バケット内の共通の場所に、すべてのインベントリリストファイルをグループ化できます。
- *source-bucket* は、インベントリリストの対象であるソースバケットです。複数の異なるソースバケットから複数のインベントリレポートが同じ保存先バケットに送信されたときの競合を避けるには、ソースバケット名を追加します。
- 同じソースバケットから同じ保存先バケットに複数のインベントリレポートが送信されたときの競合を避けるには、*config-ID* を追加します。*config-ID* は、インベントリレポートの設定から取得されるため、設定時に定義したレポートの名前になります。

- *YYYY-MM-DDTHH-MMZ* は、インベントリのレポート生成プロセスでバケットのスキャンを開始した日時を示すタイムスタンプです (例: 2016-11-06T21-32Z)。
- manifest.json はマニフェストファイルです。
- manifest.checksum は、manifest.json ファイルのコンテンツの MD5 ハッシュです。
- symlink.txt は Apache Hive 互換のマニフェストファイルです。

インベントリリストは、保存先バケットの次の場所に、毎日または毎週発行されます。

```
destination-prefix/source-bucket/config-ID/data/example-file-name.csv.gz
...
destination-prefix/source-bucket/config-ID/data/example-file-name-1.csv.gz
```

- *destination-prefix* は、インベントリ設定でオプションで指定するオブジェクトキー名プレフィックスです。このプレフィックスを使用して、保存先バケットの共通の場所にすべてのインベントリリストファイルをグループ化できます。
- *source-bucket* は、インベントリリストの対象であるソースバケットです。複数の異なるソースバケットから複数のインベントリレポートが同じ保存先バケットに送信されたときの競合を避けるには、ソースバケット名を追加します。
- *example-file-name*.csv.gz は、CSV インベントリファイルの 1 つです。ORC インベントリ名はファイル名拡張子 .orc で終わり、Parquet インベントリ名はファイル名拡張子 .parquet で終わります。

  インベントリリストファイルは、Amazon S3 Select を使用してクエリできます。Amazon S3 コンソールで、インベントリリストの名前 (例: *destination-prefix*/*source-bucket*/*config-ID*/data/*example-file-name.csv.gz*) を選択します。次に、[オブジェクトアクション]、[S3 Select を使用したクエリ] の順に選択します。S3 Select 集計関数を使用してインベントリリストファイルをクエリする方法の例については、「SUM の例」を参照してください。

インベントリマニフェスト

マニフェストファイルの manifest.json と symlink.txt は、インベントリファイルの場所を記述します。新しいインベントリリストが配信されるたびに、新しいセットのマニフェストファイルが作成されます。これらのファイルは互いに上書きされる可能性があります。バージョニングが有効なバケットには、Amazon S3 はマニフェストファイルの新しいバージョンを作成します。

manifest.json ファイルに含まれる各マニフェストには、メタデータおよびその他のインベントリに関する基本的な情報が記載されています。この情報には以下が含まれます。

- ソースバケット名
- 保存先バケット名
- インベントリのバージョン
- インベントリのレポート生成プロセスでバケットのスキャンを開始した日時をエポック日付形式で示す、作成タイムスタンプ
- インベントリファイルの形式とスキーマ
- 保存先バケット内に存在するインベントリファイルのリスト

manifest.json ファイルを書き込むたびに、manifest.checksum ファイルのコンテンツの MD5 ハッシュとして manifest.json ファイルが添付されます。

Example **manifest.json** ファイル内のインベントリマニフェスト

manifest.json ファイルに含まれる、CSV、ORC、Parquet 形式インベントリ用のインベントリマニフェストの例を以下に示します。

CSV

CSV 形式のインベントリの manifest.json ファイルに含まれるマニフェストの例を次に示します。

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-inventory-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "CSV",
    "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker,
 Size, LastModifiedDate, ETag, StorageClass, IsMultipartUploaded,
 ReplicationStatus, EncryptionStatus, ObjectLockRetainUntilDate, ObjectLockMode,
 ObjectLockLegalHoldStatus, IntelligentTieringAccessTier, BucketKeyStatus,
 ChecksumAlgorithm, ObjectAccessControlList, ObjectOwner",
    "files": [
        {
            "key": "Inventory/example-source-bucket/2016-11-06T21-32Z/
files/939c6d46-85a9-4ba8-87bd-9db705a579ce.csv.gz",
```

```
            "size": 2147483647,
            "MD5checksum": "f11166069f1990abeb9c97ace9cdfabc"
        }
    ]
}
```

ORC

ORC 形式のインベントリの manifest.json ファイルに含まれるマニフェストの例を次に示します。

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "ORC",
    "fileSchema":
 "struct<bucket:string,key:string,version_id:string,is_latest:boolean,is_delete_marker:boole
    "files": [
        {
            "key": "inventory/example-source-bucket/data/
d794c570-95bb-4271-9128-26023c8b4900.orc",
            "size": 56291,
            "MD5checksum": "5925f4e78e1695c2d020b9f6eexample"
        }
    ]
}
```

Parquet

manifest.json ファイルに含まれる Parquet 形式のインベントリ用のマニフェストの例を次に示します。

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "Parquet",
    "fileSchema": "message s3.inventory { required binary bucket (UTF8);
 required binary key (UTF8); optional binary version_id (UTF8); optional boolean
 is_latest; optional boolean is_delete_marker; optional int64 size; optional
```

```
 int64 last_modified_date (TIMESTAMP_MILLIS); optional binary e_tag (UTF8);
 optional binary storage_class (UTF8); optional boolean is_multipart_uploaded;
 optional binary replication_status (UTF8); optional binary encryption_status
 (UTF8); optional int64 object_lock_retain_until_date (TIMESTAMP_MILLIS); optional
 binary object_lock_mode (UTF8); optional binary object_lock_legal_hold_status
 (UTF8); optional binary intelligent_tiering_access_tier (UTF8); optional binary
 bucket_key_status (UTF8); optional binary checksum_algorithm (UTF8); optional
 binary object_access_control_list (UTF8); optional binary object_owner (UTF8);}",
    "files": [
        {
           "key": "inventory/example-source-bucket/data/
d754c470-85bb-4255-9218-47023c8b4910.parquet",
            "size": 56291,
            "MD5checksum": "5825f2e18e1695c2d030b9f6eexample"
        }
    ]
}
```

`symlink.txt` ファイルは、Hive がインベントリファイルおよび関連データファイルを自動的に検出できるようにする Apache Hive 互換のマニフェストファイルです。Hive 互換のマニフェストは、Athena や Amazon Redshift Spectrum など、Hive 互換のサービスで動作します。また、Presto、Apache Hive、Apache Spark など、多くの Hive 互換のアプリケーションでも動作します。

> **Important**
>
> `symlink.txt` Apache Hive 互換のマニフェストファイルは現在、AWS Glue では動作しません。
> Apache Hive および Apache Spark による `symlink.txt` ファイルの読み取りは、ORC 形式および Parquet 形式のインベントリファイルではサポートされていません。

Amazon Athena で Amazon S3 インベントリをクエリする

Athena を使用できるすべてのリージョンで、Amazon Athena で標準 SQL クエリを実行し、Amazon S3 インベントリのファイルをクエリできます。使用可能な AWS リージョン を確認するには、AWS リージョン 表を参照してください。

Athena は、Apache Optimized Row Columnar (ORC)、Apache Parquet、またはカンマ区切りの値 (CSV) 形式で Amazon S3 インベントリファイルをクエリできます。Athena を使用してインベント

リファイルをクエリする場合は、ORC 形式または Parquet 形式のインベントリファイルを使用することをお勧めします。ORC 形式および Parquet 形式は、より高速なクエリパフォーマンスとより低いクエリコストを提供します。ORC および Parquet は、Apache Hadoop 向けに設計された自己記述型、型認識型の列指向ファイル形式です。列形式の場合、リーダーは現在のクエリに必要な列だけを読み取り、圧縮解除し、処理することができます。Amazon S3 インベントリの ORC 形式および Parquet 形式は、すべての AWS リージョンで使用できます。

Athena を使用して Amazon S3 インベントリファイルをクエリするには

1. Athena テーブルを作成します。テーブルの作成の詳細については、Amazon Athena ユーザーガイドの「 Amazon Athena でのテーブルの作成 」を参照してください。
2. クエリ対象のインベントリレポートが ORC 形式、Parquet 形式、CSV 形式のいずれであるかによって、以下のサンプルクエリテンプレートのいずれかを使用してクエリを作成します。

   - Athena を使用して ORC 形式のインベントリレポートをクエリするときは、次のサンプルクエリをテンプレートとして使用します。

     次のサンプルクエリには、ORC 形式のインベントリレポートのすべてのオプションフィールドが含まれています。

     このサンプルクエリを使用するには、以下を実行します。

     - *your_table_name* を、作成した Athena テーブルの名前に置き換えます。
     - インベントリ用に選択しなかったすべてのオプションフィールドを削除し、クエリとインベントリ用に選択したフィールドを対応させます。
     - 設定に応じて、次のバケット名とインベントリの場所 (設定 ID) を置き換えます。

       s3://*DOC-EXAMPLE-BUCKET*/*config-ID*/hive/

     - projection.dt.range の *2022-01-01-00-00* 日付を、Athena でデータをパーティション化する時間範囲の最初の日に置き換えます。詳細については、「Athena でのデータのパーティション化」を参照してください。

```
CREATE EXTERNAL TABLE your_table_name(
         bucket string,
         key string,
         version_id string,
         is_latest boolean,
         is_delete_marker boolean,
         size bigint,
         last_modified_date timestamp,
```

```
          e_tag string,
          storage_class string,
          is_multipart_uploaded boolean,
          replication_status string,
          encryption_status string,
          object_lock_retain_until_date bigint,
          object_lock_mode string,
          object_lock_legal_hold_status string,
          intelligent_tiering_access_tier string,
          bucket_key_status string,
          checksum_algorithm string,
          object_access_control_list string,
          object_owner string
) PARTITIONED BY (
        dt string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
  STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
  LOCATION 's3://source-bucket/config-ID/hive/'
  TBLPROPERTIES (
    "projection.enabled" = "true",
    "projection.dt.type" = "date",
    "projection.dt.format" = "yyyy-MM-dd-HH-mm",
    "projection.dt.range" = "2022-01-01-00-00,NOW",
    "projection.dt.interval" = "1",
    "projection.dt.interval.unit" = "HOURS"
  );
```

- Athena を使用して Parquet 形式のインベントリレポートをクエリする場合は、ORC 形式のレポートでサンプルクエリを使用します。ただし、`ROW FORMAT SERDE` ステートメントで ORC SerDe の代わりに次の Parquet SerDe を使用します。

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
```

- Athena を使用して CSV 形式のインベントリレポートをクエリするときは、次のサンプルクエリをテンプレートとして使用します。

  次のサンプルクエリには、CSV 形式のインベントリレポートのすべてのオプションフィールドが含まれています。

  このサンプルクエリを使用するには、以下を実行します。

- *your_table_name* を、作成した Athena テーブルの名前に置き換えます。
- インベントリ用に選択しなかったすべてのオプションフィールドを削除し、クエリとインベントリ用に選択したフィールドを対応させます。
- 設定に応じて、次のバケット名とインベントリの場所 (設定 ID) を置き換えます。

  s3://*DOC-EXAMPLE-BUCKET*/*config-ID*/hive/

- projection.dt.range の *2022-01-01-00-00* 日付を、Athena でデータをパーティション化する時間範囲の最初の日に置き換えます。詳細については、「Athena でのデータのパーティション化」を参照してください。

```
CREATE EXTERNAL TABLE your_table_name(
         bucket string,
         key string,
         version_id string,
         is_latest boolean,
         is_delete_marker boolean,
         size string,
         last_modified_date string,
         e_tag string,
         storage_class string,
         is_multipart_uploaded boolean,
         replication_status string,
         encryption_status string,
         object_lock_retain_until_date string,
         object_lock_mode string,
         object_lock_legal_hold_status string,
         intelligent_tiering_access_tier string,
         bucket_key_status string,
         checksum_algorithm string,
         object_access_control_list string,
         object_owner string
) PARTITIONED BY (
        dt string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
  STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
  LOCATION 's3://source-bucket/config-ID/hive/'
  TBLPROPERTIES (
    "projection.enabled" = "true",
    "projection.dt.type" = "date",
```

```
    "projection.dt.format" = "yyyy-MM-dd-HH-mm",
    "projection.dt.range" = "2022-01-01-00-00,NOW",
    "projection.dt.interval" = "1",
    "projection.dt.interval.unit" = "HOURS"
  );
```

3. 以下の例に示すように、インベントリに対してさまざまなクエリを実行できるようになりました。*user input placeholder* を、ユーザー自身の情報に置き換えます。

```
# Get a list of the latest inventory report dates available.
SELECT DISTINCT dt FROM your_table_name ORDER BY 1 DESC limit 10;

# Get the encryption status for a provided report date.
SELECT encryption_status, count(*) FROM your_table_name WHERE dt = 'YYYY-MM-DD-HH-
MM' GROUP BY encryption_status;

# Get the encryption status for inventory report dates in the provided range.
SELECT dt, encryption_status, count(*) FROM your_table_name
WHERE dt > 'YYYY-MM-DD-HH-MM' AND dt < 'YYYY-MM-DD-HH-MM' GROUP BY dt,
 encryption_status;
```

オブジェクトアクセスコントロールリスト (オブジェクト ACL) フィールドをインベントリレポートに追加するように S3 インベントリを設定すると、レポートにはオブジェクト ACL フィールドの値が base64 エンコードされた文字列として表示されます。オブジェクト ACL フィールドのデコードされた値を JSON で取得するには、Athena を使用してこのフィールドをクエリできます。以下のクエリの例を参照してください。オブジェクト ACL フィールドの詳細については、「オブジェクト ACL フィールドの使用」を参照してください。

```
# Get the S3 keys that have Object ACL grants with public access.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'uri') = 'http://acs.amazonaws.com/groups/global/AllUsers'
```

```
# Get the S3 keys that have Object ACL grantees in addition to the object owner.
WITH grants AS
    (SELECT key,
    from_utf8(from_base64(object_access_control_list)) AS
 object_access_control_list,
         object_owner,
         CAST(json_extract(from_utf8(from_base64(object_access_control_list)),
         '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))) AS grants_array
    FROM your_table_name)
SELECT key,
       grant,
       objectowner
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE cardinality(grants_array) > 1 AND element_at(grant, 'canonicalId') !=
 object_owner;
```

```
# Get the S3 keys with READ permission that is granted in the Object ACL.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'permission') = 'READ';
```

```
# Get the S3 keys that have Object ACL grants to a specific canonical user ID.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
```

```
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'canonicalId') = 'user-canonical-id';
```

```
# Get the number of grantees on the Object ACL.
SELECT key,
       object_access_control_list,
       json_array_length(json_extract(object_access_control_list,'$.grants')) AS
 grants_count
FROM your_table_name;
```

Athena の詳しい使用方法については、「Amazon Athena ユーザーガイド」を参照してください。

Amazon S3 インベントリレポートの空のバージョン ID 文字列を NULL 文字列に変換します。

> **Note**
>
> 以下の手順は、すべてのバージョンを含む Amazon S3 インベントリレポートにのみ適用されます。また、「すべてのバージョン」レポートが S3 バージョニングが有効になっているバケットの S3 バッチオペレーションのマニフェストとして使用される場合に限ります。現在のバージョンのみを指定する S3 インベントリレポートの文字列を変換する必要もありません。

S3 インベントリレポートは、S3 バッチ操作のマニフェストとして使用できます。ただし、バケットで S3 バージョニングが有効になっている場合、すべてのバージョンを含む S3 インベントリレポートはバージョン ID フィールドに空の文字列で NULL バージョン対応オブジェクトをマークします。インベントリレポートにすべてのオブジェクトバージョン ID が含まれる場合、バッチ操作は `null` 文字列をバージョン ID として使用しますが、空の文字列ではありません。

S3 バッチ操作ジョブが「すべてのバージョン」S3 インベントリレポートをマニフェストとして使用すると、バージョン ID フィールドに空の文字列を含むオブジェクトのすべてのタスクが失敗しま

す。S3 インベントリレポートのバージョン ID フィールドの空文字列をバッチ操作用の null 文字列に変換するには、次の手順を使用します。

バッチ操作で使用するために Amazon S3 インベントリレポートを更新します。

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. S3 インベントリレポートに移動します。インベントリレポートの構成中に指定した宛先バケットにインベントリレポートを配置します。インベントリレポートの場所の詳細については、インベントリリストの検索 を参照してください。

   a. 宛先バケットを選択します。
   b. フォルダを選択します。フォルダの名前は、元のソースバケットにちなんで付けられます。
   c. インベントリ設定の名前が付いたフォルダを選択します。
   d. hive という名前のフォルダの横にあるチェックボックスを選択します。ページの上部で、[S3 URI をコピーする] を選択し、フォルダの S3 URI をコピーします。
3. https://console.aws.amazon.com/athena/ で Amazon Athena コンソールを開きます。
4. クエリエディタで、[設定] を選択し、[管理] を選択します。リポジトリの [設定の管理] ページのクエリ結果の場所 で、クエリ結果を保存する S3 バケットを選択します。
5. クエリエディタで、次のコマンドを使用して、インベントリレポートのデータを保持する Athena テーブルを作成します。任意の名前 *table_name* に置き換え、LOCATION 節に、以前にコピーした S3 URI を挿入します。その後、[Run] を選択してクエリを実行します。

```
CREATE EXTERNAL TABLE table_name(bucket string, key string,
 version_id string) PARTITIONED BY (dt string)ROW FORMAT SERDE
 'org.apache.hadoop.hive.serde2.OpenCSVSerde'STORED AS INPUTFORMAT
 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat' OUTPUTFORMAT
 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat' LOCATION 'Copied S3 URI';
```

6. クエリエディタをクリアするには、[Clear] を選択します。次のコマンドを使用して、インベントリレポートをテーブルにロードします。前のステップで選択したもの *table_name* に置き換えてください。その後、[Run] を選択してクエリを実行します。

```
MSCK REPAIR TABLE table_name;
```

7. クエリエディタをクリアするには、[Clear] を選択します。以下の SELECT クエリを実行して、元のインベントリレポートのすべてのエントリを取得し、空のバージョン ID を null 文字列に置き換えます。先に選択したもの *table_name* に置き換え、節内 *YYYY-MM-DD-HH-MM* をこの

ツールを実行させたいインベントリー・レポートの日付 WHERE に置き換えます。次に [Run] を選択して、クエリを実行します。

```
SELECT bucket as Bucket, key as Key, CASE WHEN version_id = '' THEN 'null' ELSE
 version_id END as VersionId FROM table_name WHERE dt = 'YYYY-MM-DD-HH-MM';
```

8. Amazon S3コンソール (https://console.aws.amazon.com/s3/) に戻り、先ほど クエリ結果の場所 として選択した S3 バケットに移動します。内部には、日付で終わる一連のフォルダがあるはずです。

   例えば、s3://***DOC-EXAMPLE-BUCKET***/***query-result-location***/Unsaved/2021/10/07/ のようなものがあるはずです。実行した .csv クエリの結果を含む SELECT ファイルが表示されるはずです。

   最新の更新日の CSV ファイルを選択してください。次のステップのために、このファイルをローカル・マシンにダウンロードします。

9. 生成された CSV ファイルには、ヘッダー行が含まれています。この CSV ファイルを S3 バッチ操作ジョブの入力として使用するには、ヘッダー行を削除する必要があります。これは、バッチ操作では CSV マニフェストのヘッダー行をサポートしないためです。

   ヘッダー行を削除するには、ファイルで次のいずれかのコマンドを実行します。*file.csv* を CSV ファイルの名前に置き換えてください。

   macOS および Linux マシンの場合、ターミナル ウィンドウで tail コマンドを実行します。

   ```
   tail -n +2 file.csv > tmp.csv && mv tmp.csv file.csv
   ```

   Windows マシンの場合、Windows PowerShell ウィンドウで次のスクリプトを実行します。*File-location* をファイルへのパスに置き換え、*file.csv* をファイルに置き換えます。

   ```
   $ins = New-Object System.IO.StreamReader File-location\file.csv
   $outs = New-Object System.IO.StreamWriter File-location\temp.csv
   try {
       $skip = 0
       while ( !$ins.EndOfStream ) {
           $line = $ins.ReadLine();
           if ( $skip -ne 0 ) {
               $outs.WriteLine($line);
           } else {
   ```

```
            $skip = 1
        }
    }
} finally {
    $outs.Close();
    $ins.Close();
}
Move-Item File-location\temp.csv File-location\file.csv -Force
```

10. CSV ファイルからヘッダー行を削除したら、S3 バッチ操作ジョブのマニフェストとして使用する準備が整いました。CSV ファイルを S3 バケットまたは任意の場所にアップロードし、CSV ファイルをマニフェストとして使用して バッチ操作ジョブを作成します。

    バッチ操作のジョブの作成の詳細については、S3 バッチオペレーションジョブの作成 を参照してください。

オブジェクト ACL フィールドの使用

Amazon S3 インベントリレポートは、S3 ソースバケット内のオブジェクトのリストと、各オブジェクトのメタデータを表示します。オブジェクトアクセスコントロールリスト (ACL) フィールドは、Amazon S3 インベントリで使用できるメタデータフィールドです。オブジェクト ACL フィールドは、特に各オブジェクトのアクセスコントロールリスト (ACL) を示します。オブジェクトの ACL は、このオブジェクトへのアクセスを許可する AWS アカウントまたはグループと、付与するアクセスの種類を定義します。詳細については、アクセスコントロールリスト (ACL) の概要およびAmazon S3 インベントリのリストを参照してください。

Amazon S3 インベントリレポートのオブジェクト ACL フィールドは、JSON 形式で定義します。JSON データには、以下のデータが含まれます。

- `version` — インベントリレポートのオブジェクト ACL フィールド形式のバージョン。バージョンは日付形式 `yyyy-mm-dd` となります。
- `status` — 指定できる値は `AVAILABLE` または `UNAVAILABLE` です。オブジェクトでオブジェクト ACL が使用できるかどうかを示します。オブジェクト ACL のステータスが `UNAVAILABLE` である場合、インベントリレポートのオブジェクト所有者フィールドの値も `UNAVAILABLE` になります。
- `grants` — 被付与者とアクセス許可のペア。オブジェクト ACL から付与された被付与者ごとのアクセス許可のステータスを示します。被付与者に設定できる値は `CanonicalUser` および `Group` です。被付与者の詳細については、「アクセスコントロールリストの被付与者」を参照してください。

Group タイプの被付与者の場合、付与者とアクセス許可のペアには以下の属性が含まれます。

- uri — 事前定義された Amazon S3 グループ。
- permission — オブジェクトに付与される ACL アクセス許可。詳細については、「オブジェクトに対する ACL アクセス許可」を参照してください。
- type — タイプ Group。これは被付与者がグループであることを示します。

CanonicalUser タイプの被付与者の場合、付与者とアクセス許可のペアには以下の属性が含まれます。

- canonicalId — 難読化された形式の AWS アカウント ID。AWS アカウント の正規ユーザー ID は、そのアカウントに固有です。正規ユーザー ID を取得できます。詳細については、「AWS Account Management リファレンスガイド」の「AWS アカウント の正規ユーザー ID を検索する」を参照してください。

> **Note**
>
> ACL の被付与者が AWS アカウントの E メールアドレスである場合、S3 インベントリは、その AWS アカウントの canonicalId と CanonicalUser タイプを使用してこの被付与者を指定します。詳細については、「アクセスコントロールリストの被付与者」を参照してください。

- permission — オブジェクトに付与される ACL アクセス許可。詳細については、「オブジェクト対する ACL アクセス許可」を参照してください。
- type — タイプ CanonicalUser。これは、被付与者が AWS アカウントであることを示します。

次の例は、JSON 形式のオブジェクト ACL フィールドに指定できる値を示しています。

```
{
    "version": "2022-11-10",
    "status": "AVAILABLE",
    "grants": [{
        "uri": "http://acs.amazonaws.com/groups/global/AllUsers",
        "permission": "READ",
        "type": "Group"
    }, {
        "canonicalId": "example-canonical-id",
        "permission": "FULL_CONTROL",
```

```
        "type": "CanonicalUser"
    }]
}
```

> **Note**
>
> オブジェクト ACL フィールドは JSON 形式で定義されます。インベントリレポートには、オブジェクト ACL フィールドの値が base64 エンコードされた文字列として表示されます。例えば、次のオブジェクト ACL フィールドが JSON 形式で存在するとします。
>
> ```
> {
>         "version": "2022-11-10",
>         "status": "AVAILABLE",
>         "grants": [{
>             "canonicalId": "example-canonical-user-ID",
>             "type": "CanonicalUser",
>             "permission": "READ"
>         }]
> }
> ```
>
> オブジェクト ACL フィールドはエンコードされ、次の base64 エンコードされた文字列として表示されます。
>
> ```
> eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3siY2Fub25pY2FsSW
> ```
>
> オブジェクト ACL フィールドのデコードされた値を JSON で取得するには、Amazon Athena でこのフィールドをクエリできます。クエリの例については、「Amazon Athena で Amazon S3 インベントリをクエリする」を参照してください。

オブジェクトのレプリケーション

レプリケーションを使用すると、Amazon S3 バケット間でオブジェクトを自動で非同期的にコピーできます。オブジェクトのレプリケーション用に設定されたバケットは、同じ AWS アカウント が所有することも、異なるアカウントが所有することもできます。オブジェクトは、単一または複数の送信先バケットにレプリケートできます。送信先バケットは、異なる AWS リージョン でも、ソースバケットと同じリージョン内でも配置することができます。

レプリケーションには、ライブレプリケーションとオンデマンドレプリケーションの 2 種類があります。

- ライブレプリケーション – レプリケート元のバケットに書き込まれる際に、新しく作成されたオブジェクトまたは更新されたオブジェクトを自動的にレプリケートするには、ライブレプリケーションを使用します。ライブレプリケーションでは、レプリケーションを設定する前にバケットに存在していたオブジェクトはレプリケートされません。レプリケーションを設定する前に存在していたオブジェクトをレプリケートするには、オンデマンドレプリケーションを使用します。
- オンデマンドレプリケーション — レプリケート元のバケットから 1 つ以上のレプリケート先バケットにオンデマンドで既存のオブジェクトをレプリケートするには、S3 バッチレプリケーションを使用します。既存のオブジェクトのレプリケーションの詳細については、「S3 バッチレプリケーションを使用する状況」を参照してください。

ライブレプリケーションには、クロスリージョンレプリケーション (CRR) と同一リージョンレプリケーション (SRR) の 2 つの形式があります。

- クロスリージョンレプリケーション (CRR) - 異なる AWS リージョン 内の Amazon S3 バケット間でオブジェクトをレプリケートするには、CRR を使用します。CRR の詳細については、「the section called “クロスリージョンレプリケーションを使用する場合”」を参照してください。
- 同一リージョンレプリケーション (SRR) - 同じ AWS リージョン 内の Amazon S3 バケット間でオブジェクトをコピーするには、SRR を使用します。SRR の詳細については、「the section called “同一リージョンレプリケーションを使用する時”」を参照してください。

トピック

- レプリケーションメトリクスと S3 イベント通知による、進捗状況のモニタリング
- S3 バッチレプリケーションを使用した既存のオブジェクトのレプリケーション

レプリケーションを使用する理由

レプリケーションは、以下の場合に役立ちます。

- メタデータを保持しながらオブジェクトをレプリケートする — レプリケーションを使用すると、元のオブジェクトの作成時刻やバージョン ID などのすべてのメタデータを保持するオブジェクトのコピーを作成できます。この機能は、レプリカがレプリケート元オブジェクトと同じであることを確認する必要がある場合に重要です。
- 別のストレージクラスにオブジェクトをレプリケートする – レプリケーションを使用して、オブジェクトを S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive、またはレプリケート先バケットの別のストレージクラスに直接配置できます。データを同じストレージクラスにレプリケートし、レプリケート先バケットのライフサイクル設定を使用して、オブジェクトが古くなるにつれてより古いストレージクラスに移動させることもできます。
- オブジェクトのコピーを別の所有権で保持する — レプリケート元オブジェクトの所有者に関係なく、レプリカの所有権をレプリケート先バケット所有者である AWS アカウント に変更するように Simple Storage Service (Amazon S3) に指示できます。これは 所有者オーバーライド オプションと呼ばれます。このオプションを使用すると、オブジェクトのレプリカへのアクセスを制限できます。
- 複数の AWS リージョンに格納されたオブジェクトを保持する – 異なる AWS リージョン間で複数のレプリケート先バケットを設定して、データの保存場所を地理的に異なる場所にします。この機能は、特定のコンプライアンス要件を満たすのに役立つことがあります。
- 15 分以内にオブジェクトをレプリケート — S3 Replication Time Control (S3 RTC) を使用して、予測可能な時間枠内で、同じ AWS リージョンまたは異なるリージョン間でデータをレプリケートできます。S3 RTC は、Simple Storage Service (Amazon S3) 内に保存されている新規オブジェクトの 99.99% を 15 分以内にレプリケートします (サービスレベルアグリーメントに基づく)。詳細については、「the section called “S3 Replication Time Control の有効化”」を参照してください。

> **Note**
>
> S3 RTC はバッチレプリケーションには適用されません。バッチレプリケーションはオンデマンドレプリケーションジョブで、S3 バッチオペレーションで追跡できます。詳細については、「ジョブステータスと完了レポートの追跡」を参照してください。

- バケットの同期、既存オブジェクトのレプリケート、以前に失敗したオブジェクトまたはレプリケートされたオブジェクトのレプリケート - バケットを同期して既存のオブジェクトをレプリケートするには、オンデマンドレプリケーションアクションとしてバッチレプリケーションを使用します。バッチレプリケーション使用時の詳細については、「S3 バッチレプリケーションを使用する状況」を参照してください。
- オブジェクトをレプリケートし、別のAWS リージョン のバケットにフェイルオーバーする - データレプリケーション中にバケット間ですべてのメタデータとオブジェクトを同期させるために、Amazon S3 マルチリージョンアクセスポイントのフェイルオーバーコントロールを構成する前に双方向レプリケーションルールを使用します。双方向のレプリケーションルールにより、トラフィックがフェイルオーバーする S3 バケットにデータが書き込まれると、そのデータがソースバケットにレプリケートされます。

クロスリージョンレプリケーションを使用する場合

異なる AWS リージョン 内の Amazon S3 バケット間でオブジェクトをコピーするには、S3 クロスリージョンレプリケーション (CRR) を使用します。CRR は、次の場合に役立ちます。

- コンプライアンス要件を満たす — Simple Storage Service (Amazon S3) はデフォルトで地理的に離れた複数のアベイラビリティーゾーンにデータを保存しますが、コンプライアンス要件によっては、さらに離れた場所にデータを保存することが要求される場合があります。これらの要件を満たすには、遠く離れた AWS リージョン にデータをレプリケートするクロスリージョンレプリケーションを使用します。
- レイテンシーを最小にする — ユーザーが地理的に離れた 2 つの場所にいる場合、ユーザーにより近い AWS リージョン にオブジェクトのコピーを保持することで、オブジェクトにアクセスする際のレイテンシーを最小にすることができます。
- オペレーション効率を向上する — コンピューティングクラスターが 2 つの異なる AWS リージョン にあり、同じオブジェクトセットを分析している場合、これら 2 つのリージョンにオブジェクトのコピーを保持できます。

同一リージョンレプリケーションを使用する時

同一 AWS リージョン 内の Amazon S3 バケット間でオブジェクトをコピーするには、同一リージョンレプリケーション (SRR) を使用します。SRR は、次の場合に役立ちます。

- ログを 1 つのバケットに集約する — 複数のバケットまたは複数のアカウントにログを保存している場合、ログを 1 つのリージョン内バケットに簡単にレプリケートできます。そうすることで、ログを一箇所でよりシンプルに処理できます。
- 本番稼働用アカウントとテストアカウント間のライブレプリケーションを設定する — お客様またはお客様のユーザーが保持する本稼働用アカウントとテストアカウントで同じデータを使用する場合、オブジェクトメタデータを維持しながら、これらの複数のアカウント間でオブジェクトをレプリケートできます。
- データ主権法に準拠する — データの複数のコピーを特定のリージョン内の複数の異なる AWS アカウントに保存することを義務付けられる場合があります。同一リージョンレプリケーションを使用すると、コンプライアンス規制によりデータを国外に持ち出すことが許可されていない場合に、重要なデータを自動的にレプリケートできます。

双方向レプリケーションを使用する場合

- 複数の AWS リージョン で共有するデータセットを構築する — レプリカ変更の同期により、オブジェクトのアクセスコントロールリスト (ACL)、オブジェクトタグ、またはオブジェクトロックなどのメタデータの変更を、レプリケーションオブジェクトで簡単に複製することができます。この双方向のレプリケーションは、すべてのオブジェクトとオブジェクトメタデータの変更を同期させたい場合に重要です。同一または異なる AWS リージョン にある 2 つ以上のバケット間で双方向レプリケーションを実行する場合、新規または既存のレプリケーションルールで[レプリカ変更の同期を有効にする]ことができます。
- フェイルオーバー中にデータをリージョン間で同期させる — マルチリージョンアクセスポイントから直接 S3 クロスリージョンレプリケーション (CRR) で双方向レプリケーションルールを設定することで、AWS リージョン 間のバケットのデータを同期させることができます。フェイルオーバーを開始するタイミングについて十分な情報に基づいた決定を行うために、S3 レプリケーションメトリクスを有効にして Amazon CloudWatch 内、S3 Replication Time Control (S3 RTC) 内、またはマルチリージョンアクセスポイントからのレプリケーションをモニタリングすることもできます。
- アプリケーションの可用性を高める — リージョンのトラフィックが中断した場合でも、双方向のレプリケーションルールを使用して、データレプリケーション中にすべてのメタデータとオブジェクトをバケット間で同期させることができます。

S3 バッチレプリケーションを使用する状況

バッチレプリケーションは、オンデマンドオプションとして、既存のオブジェクトを異なるバケットにレプリケートします。ライブレプリケーションとは異なり、これらのジョブは必要に応じて実行できます。バッチレプリケーションは、以下の場合に役立ちます。

- 既存のオブジェクトのレプリケーション — バッチレプリケーションを使用して、同じリージョンレプリケーションまたはクロスリージョンレプリケーションが設定される前に、バケットに追加されたオブジェクトをレプリケートできます。
- 以前にレプリケートに失敗したオブジェクトをレプリケートする - バッチレプリケーションジョブをフィルタリングして、レプリケーションステータスが [FAILED] (失敗) のオブジェクトをレプリケートできます。
- すでにレプリケートされたオブジェクトをレプリケートする — データの複数のコピーを別々の AWS アカウントまたは AWS リージョンに保存することを義務付けられる場合があります。バッチレプリケーションでは、新規に追加された宛先に既存のオブジェクトをレプリケートできます。
- レプリケーションルールから作成されたオブジェクトのレプリカをレプリケートする — レプリケーション設定では、レプリケート先バケットにオブジェクトのレプリカが作成されます。オブジェクトのレプリカは、バッチレプリケーションでのみレプリケートできます。

ワークロードの要件とライブレプリケーション

ワークロードの要件に応じて、一部のレプリケーションタイプは他のレプリケーションよりもユースケースに適していることがあります。次の表を使用して、使用するレプリケーションのタイプと、ワークロードで S3 Replication Time Control (S3 RTC) を使用するかどうかを判断します。S3 RTC は、Amazon S3 内に保存されている新規オブジェクトの 99.99% を 15 分以内にレプリケートします (サービスレベルアグリーメント (SLA) に基づく)。詳細については、「the section called “S3 Replication Time Control の有効化”」を参照してください。

ワークロードの要件に基づくレプリケーションの比較

ワークロードの要件	S3 RTC (15 分の SLA)	クロスリージョンレプリケーション (CRR)	同一リージョンレプリケーション (SRR)
異なる AWS アカウント 間でオブジェクトをレプリケートする	はい	はい	はい

ワークロードの要件	S3 RTC (15 分の SLA)	クロスリージョンレプリケーション (CRR)	同一リージョンレプリケーション (SRR)
24 ~ 48 時間以内に同じ AWS リージョン 内のオブジェクトをレプリケートする (SLA に基づかない)	いいえ	いいえ	はい
24 ~ 48 時間以内に異なる AWS リージョン 内のオブジェクトをレプリケートする (SLA に基づかない)	いいえ	はい	いいえ
予測可能なレプリケーション時間: SLA に基づき、15 分以内にオブジェクトの 99.9% をレプリケートする	はい	いいえ	いいえ

Amazon S3 がレプリケートするもの

Amazon S3 は、レプリケーションに設定されているバケット内の特定のアイテムのみをレプリケートします。

トピック

- レプリケーション設定でレプリケートされるものは何ですか?
- レプリケーション設定でレプリケートされないものは何ですか?
- バケットのデフォルトの暗号化がレプリケーションに与える影響

レプリケーション設定でレプリケートされるものは何ですか?

デフォルトで、Simple Storage Service (Amazon S3) は以下をレプリケートします。

- レプリケーション設定の追加後に作成されたオブジェクト。
- 暗号化されていないオブジェクト
- ユーザー提供のキー (SSE-C) を使用して暗号化されたオブジェクト、Amazon S3 マネージドキー (SSE-S3) の下で保管時に暗号化されたオブジェクト、および AWS Key Management Service (SSE-KMS) に保存されている KMS キーで暗号化されたオブジェクト。詳細については、「the section called “暗号化オブジェクトのレプリケート”」を参照してください。
- レプリケート元オブジェクトからレプリカへのオブジェクトメタデータ。レプリカからレプリケート元オブジェクトへのメタデータのレプリケーションについては、Amazon S3 レプリカの変更同期によるメタデータ変更のレプリケートを参照してください。
- バケット所有者がオブジェクトとアクセスコントロールリスト (ACL) の読み取り権限を持つ、レプリケート元バケットのオブジェクトのみをレプリケートします。

  リソース所有者の詳細については「Amazon S3 のバケットとオブジェクトの所有権」を参照してください。
- レプリケート元バケットとレプリケート先バケットが同じアカウントによって所有されていない場合のレプリカの所有権の変更を Amazon S3 に指示しない限り、オブジェクト ACL は更新されます。

  詳細については、「レプリカ所有者の変更」を参照してください。

  Amazon S3 が 2 つの ACL を同期させるまでしばらく時間がかかる可能性があります。オーナーシップのこの変更は、バケットにレプリケーション設定を追加した後に作成されたオブジェクトにのみ適用されます。
- オブジェクトタグ、存在する場合。
- S3 オブジェクトロックの保持情報 (ある場合)。

  Amazon S3 で保持情報が適用されているオブジェクトをレプリケートすると、それらと同じ保持制御がレプリカに適用され、レプリケート先バケットに設定されているデフォルトの保持期間は上書きされます。レプリケート元バケット内のオブジェクトに保持制御が適用されておらず、デフォルトの保持期間が設定されているレプリケート先バケットにレプリケートすると、レプリケート先バケットのデフォルトの保持期間がオブジェクトのレプリカに適用されます。詳細については、「S3 オブジェクトロックの使用」を参照してください。

削除オペレーションがレプリケーションに与える影響

レプリケート元バケットからオブジェクトを削除すると、デフォルトで次のアクションが実行されます。

- オブジェクトバージョン ID を指定せずに DELETE リクエストを行った場合、Amazon S3 は削除マーカーを追加します。Amazon S3 では、削除マーカーを次のように扱います。
  - 最新バージョンのレプリケーション設定を使用している (すなわち、レプリケーション設定ルールで`Filter`要素を指定している) 場合、Amazon S3 は削除マーカーをレプリケートしません。ただし、タグベース以外のルールには削除マーカーレプリケーションを追加できます。詳細については、「バケット間での削除マーカーのレプリケーション」を参照してください。
  - `Filter`要素を指定しない場合、Amazon S3 はレプリケーション設定がバージョン V1 であるとみなし、ユーザーアクションの結果として生じた削除マーカーをレプリケートします。ただし、ライフサイクルアクションによって Amazon S3 がオブジェクトを削除した場合、削除マーカーはレプリケート先のバケットにはレプリケートされません。
- DELETE リクエストで削除するオブジェクトバージョン ID を指定した場合、Amazon S3 はレプリケート元バケット内のそのオブジェクトバージョンを削除します。しかし、レプリケート先バケット内でその削除をレプリケートすることはありません。これは、レプリケート先バケットからは、同じオブジェクトバージョンを削除しないことを意味します。これは悪意のある削除からデータを保護します。

レプリケーション設定でレプリケートされないものは何ですか?

デフォルトで、Simple Storage Service (Amazon S3) は以下をレプリケートしません。

- 別のレプリケーションルールによって作成されたレプリカである、レプリケート元バケットのオブジェクト。例えば、バケット A がレプリケート元でバケット B がレプリケート先であるレプリケーションを設定するとします。ここで、バケット B をレプリケート元、バケット C をレプリケート先とする別のレプリケーション設定を追加したとします。この場合、バケット A のオブジェクトのレプリカであるバケット B のオブジェクトは、バケット C にレプリケートされません。

  レプリカであるオブジェクトをレプリケートするには、バッチレプリケーションを使用します。バッチレプリケーションの設定の詳細については、「既存のオブジェクトのレプリケーション」を参照してください。
- 既に別のレプリケート先にレプリケートされている、レプリケート元バケット内のオブジェクト。例えば、既存のレプリケーション設定でレプリケート先バケットを変更した場合、Simple Storage Service (Amazon S3) がそのオブジェクトを再度レプリケートすることはありません。

  以前にレプリケートされたオブジェクトをレプリケートするには、バッチレプリケーションを使用します。バッチレプリケーションの設定の詳細については、「既存のオブジェクトのレプリケーション」を参照してください。

- バッチレプリケーションでは、レプリケート先バケットからオブジェクトのバージョン ID で削除されたオブジェクトの再レプリケーションはサポートされません。これらのオブジェクトを再レプリケートするには、バッチコピージョブを使用してソースオブジェクトを所定の場所でコピーします。これらのオブジェクトを所定の場所でコピーすると、レプリケート元バケットにオブジェクトの新しいバージョンが作成され、レプリケート先へのレプリケーションが自動的に開始されます。バッチコピーの使用方法については、「バッチ操作を使用してオブジェクトをコピーする例」を参照してください。
- デフォルトでは、異なる AWS アカウント からレプリケートする場合、レプリケート元バケットに追加された削除マーカーはレプリケートされません。

  削除マーカーの複製方法については、「バケット間での削除マーカーのレプリケーション」を参照してください。
- S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive、S3 Intelligent-Tiering Archive Access、または S3 Intelligent-Tiering Deep Archive Access のストレージクラスまたはストレージ階層に保存されているオブジェクト。これらのオブジェクトは、復元して別のストレージクラスにコピーするまで複製できません。

  S3 Glacier Flexible Retrieval と S3 Glacier Deep Archive の詳細については、「アクセス頻度の低いオブジェクトのストレージクラス」を参照してください。

  S3 Intelligent-Tiering の詳細については、「Amazon S3 Intelligent-Tiering」を参照してください。
- バケット所有者が十分な許可を持っていないレプリケート元バケット内のオブジェクト。

  オブジェクト所有者がバケット所有者にアクセス許可を付与する方法の詳細については、「バケット所有者はフルコントロール権限を持ちながら、オブジェクトをアップロードするためのクロスアカウントアクセス許可を付与する」を参照してください。
- バケットレベルのサブリソースの更新

  たとえば、ライフサイクル設定を変更したり、レプリケート元バケットに通知設定を追加した場合、これらの変更はレプリケート先バケットには適用されません。この機能により、レプリケート元バケットとレプリケート先バケットで異なる設定を指定できます。
- ライフサイクル設定によって実行されたアクション。

  たとえば、ライフサイクル設定がレプリケート元バケットでのみ有効である場合、Amazon S3 は失効したオブジェクトに削除マーカーを作成しますが、その削除マーカーはレプリケートされません。同じライフサイクル設定をレプリケート元バケットとレプリケート先バケットの両方に適用

する場合は、両方で同じライフサイクル設定を有効にします。ライフサイクル設定についての詳細は、ストレージのライフサイクルの管理 を参照してください。

- ライブレプリケーションでタグベースのレプリケーションルールを使用する場合、PutObject オペレーションで一致するレプリケーションルールタグを新しいオブジェクトにタグ付けする必要があります。そうしないと、オブジェクトはレプリケートされません。PutObject オペレーション後にオブジェクトにタグが付けされると、それらのオブジェクトもレプリケートされません。

  PutObject オペレーション後にタグ付けされたオブジェクトをレプリケートするには、S3 バッチレプリケーションを使用する必要があります。バッチレプリケーションの詳細については、「既存のオブジェクトのレプリケーション」を参照してください。

バケットのデフォルトの暗号化がレプリケーションに与える影響

レプリケーション先バケットのデフォルトの暗号化を有効にすると、以下の暗号化動作が適用されます。

- レプリケート元バケットのオブジェクトが暗号化されていない場合、レプリケート先バケットのレプリカオブジェクトはレプリケート先バケットのデフォルトの暗号化設定を使用して暗号化されます。そのため、レプリケート元のオブジェクトのエンティティタグ (ETag) はレプリカオブジェクトの ETag とは異なります。アプリケーションで ETag を使用している場合は、アプリケーションを更新して、この違いを反映する必要があります。
- レプリケート元バケット内のオブジェクトが Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、または AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS) を使用して暗号化されている場合、レプリケート先バケットのレプリカオブジェクトは、レプリケート元オブジェクトと同じタイプの暗号化を使用します。レプリケート先バケットのデフォルトの暗号化設定は使用されません。

レプリケーションの要件と考慮事項

Amazon S3 レプリケーションでは以下が必要です。

- ソースバケットの所有者は、自分のアカウントに対して送信元と送信先の AWS リージョン を有効にする必要があります。レプリケート先のバケット所有者は、自分のアカウントでレプリケート先リージョンを有効にしておく必要があります。

AWS リージョン の有効化と無効化の詳細については、AWS 全般のリファレンス の「AWS リージョン の管理」を参照してください。

- レプリケート元とレプリケート先の両方のバケットで、バージョニングを有効にする必要があります。バージョニングの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。
- Amazon S3 には、お客様に代わってレプリケート元バケットから送信先バケットにオブジェクトをレプリケートするためのアクセス許可が必要です。これらのアクセス許可の詳細については、「ライブレプリケーションのアクセス許可の設定」を参照してください。
- ソースバケット所有者がバケット内のオブジェクトを所有していない場合、オブジェクト所有者は、オブジェクトアクセスコントロールリスト (ACL) を使用して、バケット所有者に READ 権限と READ_ACP 権限を付与する必要があります。詳細については、「アクセスコントロールリスト (ACL) の概要」を参照してください。
- レプリケート元バケットで S3 オブジェクトロックが有効になっている場合は、レプリケート先バケットでも S3 オブジェクトロックが有効になっている必要があります。

  Object Lock が有効になっているバケットでレプリケーションを有効にするには、AWS Command Line Interface、REST API、または AWS SDK を使用する必要があります。一般的な的な情報については、「S3 オブジェクトロックの使用」を参照してください

  > **Note**
  >
  > レプリケーションを設定するために使用する AWS Identity and Access Management (IAM) ロールで、ソース S3 バケットに対する 2 つの新しいアクセス許可を付与する必要があります。2 つの新しいアクセス許可は `s3:GetObjectRetention` と `s3:GetObjectLegalHold` です。ロールに `s3:Get*` アクセス許可がある場合、そのロールは要件を満たしています。詳細については、「ライブレプリケーションのアクセス許可の設定」を参照してください。

詳細については、「ライブレプリケーションの設定」を参照してください。

異なる AWS アカウント がレプリケート元バケットとレプリケート先バケットを所有するクロスアカウントのシナリオでレプリケーション設定を設定している場合は、次の追加の要件が適用されます。

- レプリケート先バケットの所有者は、バケットポリシーを使用してレプリケート元バケット所有者にオブジェクトをレプリケートするためのアクセス許可を付与する必要があります。詳細については、「レプリケーション元とレプリケーション先のバケットが異なる AWS アカウントによって所有されている場合の許可の付与」を参照してください。
- レプリケート先バケットをリクエスタ支払いバケットとして設定することはできません。詳細については、「ストレージ転送と使用量のリクエスタ支払いバケットの使用」を参照してください。

レプリケーションの考慮事項

レプリケーションを設定する前に、以下を考慮してください。

トピック

- ライフサイクル設定とオブジェクトのレプリカ
- バージョニング設定とレプリケーション設定
- S3 Intelligent-Tiering での S3 レプリケーションの使用
- ログ設定とレプリケーション設定
- CRR とレプリケート先のリージョン
- S3 バッチレプリケーション
- S3 Replication Time Control

ライフサイクル設定とオブジェクトのレプリカ

Amazon S3 がオブジェクトをレプリケートするのにかかる時間は、オブジェクトのサイズによって異なります。大きなオブジェクトの場合、数時間かかることもあります。レプリカがレプリケート先バケットで使用可能になるまでにはしばらく時間がかかります。レプリカの作成には、ソースバケットに対応するオブジェクトを作成するのと同じくらいの時間がかかります。レプリケート先バケットでライフサイクル設定が有効な場合、ライフサイクルルールでは、レプリケート先バケットでレプリカが使用可能になった時間ではなく、オブジェクトの元の作成時間が優先されることに注意してください。

レプリケーション設定では、バケットのバージョニングを有効にする必要があります。バケットのバージョニングを有効にする際、以下の点に注意してください。

- オブジェクトの有効期限ライフサイクル設定がある場合は、バージョニングを有効にした後に `NonCurrentVersionExpiration` ポリシーを追加して、バージョニングを有効にする前と同じ完全な削除動作を維持する必要があります。

- 移行ライフサイクル設定がある場合は、バージョニングを有効にした後に、NonCurrentVersionTransition ポリシーの追加を検討する必要があります。

バージョニング設定とレプリケーション設定

レプリケート元バケットとレプリケート先バケットはどちらも、バケットにレプリケーションを設定するときに、バージョニングが有効になっている必要があります。レプリケート元バケットとレプリケート先バケットの両方でバージョニングを有効にして、レプリケート元バケットでレプリケーションを設定した後は、次の問題が発生します。

- レプリケート元バケットのバージョニングを無効にしようとすると、Amazon S3 はエラーを返します。レプリケート元バケットのバージョニングを無効にする前に、レプリケーション設定を削除する必要があります。
- レプリケート先バケットのバージョニングを無効にすると、レプリケーションは失敗します。レプリケート元オブジェクトのレプリケーションステータスはFAILEDです。

S3 Intelligent-Tiering での S3 レプリケーションの使用

S3 Intelligent-Tiering は、最もコスト効率の高いアクセス階層にデータを自動的に移動することで、ストレージコストを最適化するように設計されたストレージクラスです。オブジェクトのモニタリングとオートメーションに対して発生する少額の月額料金で、S3 Intelligent-Tiering はアクセスパターンをモニタリングし、あまりアクセスされていないオブジェクトをより低コストのアクセス階層へ自動的に移動させることができます。

S3 バッチレプリケーションを使用して S3 Intelligent-Tiering に保存されたオブジェクトをレプリケートしたり、CopyObject または UploadPartCopy を呼び出したりすると、アクセスが構成されます。これらの場合、コピーやレプリケーションオペレーションのソースオブジェクトは上位の階層に移動されます。

S3 Intelligent-Tiering の詳細については、「Amazon S3 Intelligent-Tiering」を参照してください。

ログ設定とレプリケーション設定

レプリケーションが有効になっているバケットに Amazon S3 がログを送信している場合、ログオブジェクトをレプリケートします。

サーバーアクセスログ (サーバーアクセスログによるリクエストのログ記録) または AWS CloudTrail ログ (AWS CloudTrail を使用した Amazon S3 API コールのログ記録) をレプリケート元やレプリ

ケート先のバケットで有効にすると、Amazon S3 はレプリケーション関連のリクエストを口グに含めます。たとえば、Amazon S3 はレプリケートする各オブジェクトを口グに記録します。

CRR とレプリケート先のリージョン

異なる AWS リージョン内の S3 バケット間でオブジェクトをコピーするには、Amazon S3 クロスリージョンレプリケーション (CRR) を使用します。レプリケート先バケットのリージョンは、ビジネスニーズまたはコストを考慮して選択することができます。例えば、リージョン間のデータ転送料金は、選択したリージョンによって異なります。

たとえば、レプリケート元バケットのリージョンとして米国東部 (バージニア北部) (us-east-1) を選択したとします。米国西部 (オレゴン) (us-west-2) をレプリケート先バケットのリージョンとして選択した場合、米国東部 (オハイオ) (us-east-2) リージョンを選択した場合より多く支払うことになります。料金の詳細については、Amazon S3 の料金の「データ転送の料金」を参照してください。

同一リージョンレプリケーション (SRR) に関連するデータ転送料金はありません。

S3 バッチレプリケーション

バッチレプリケーションの考慮事項については、「S3 バッチレプリケーションに関する考慮事項」を参照してください。

S3 Replication Time Control

S3 Replication Time Control (S3 RTC) のベストプラクティスと考慮事項については、「S3 RTC のベストプラクティスとガイドライン」を参照してください。

ライブレプリケーションの設定

> **Note**
>
> レプリケーションをセットアップする前に存在していたオブジェクトは、自動的にレプリケートされません。つまり、Amazon S3 はオブジェクトをさかのぼってレプリケートすることはしません。レプリケーション設定の前に作成されたオブジェクトをレプリケートするには、S3 バッチレプリケーションを使用します。バッチレプリケーションの設定の詳細については、「既存のオブジェクトのレプリケーション」を参照してください。

ライブレプリケーション (同一リージョンレプリケーション (SRR) またはクロスリージョンレプリケーション (CRR)) を有効にするには、レプリケート元のバケットにレプリケーション設定を追加し

ます。Amazon S3 は、この設定に基づいてオブジェクトをレプリケートします。レプリケーション設定では、以下の項目を指定する必要があります。

- レプリケート先バケット – Simple Storage Service (Amazon S3) でオブジェクトをレプリケートする先のバケット。
- レプリケートするオブジェクト – レプリケート元バケットまたはサブセット内のすべてのオブジェクトをレプリケートできます。サブセットを特定するには、キー名のプレフィックス、1 つ以上のオブジェクトタグ、またはその両方を設定で指定します。

  たとえば、キー名のプレフィックス `Tax/` の オブジェクトのみをレプリケートするようにレプリケーションルールを設定した場合、Amazon S3 は `Tax/doc1` や `Tax/doc2` などのキーを持つオブジェクトをレプリケートします。しかし、`Legal/doc3` というキーを持つオブジェクトはレプリケートしません。プレフィックスと 1 つ以上のタグの両方を指定した場合、Simple Storage Service (Amazon S3) は特定のキープレフィックスとタグを持つオブジェクトのみをレプリケートします。
- AWS Identity and Access Management (IAM) ロール – Amazon S3 は、ユーザーに代わってオブジェクトをレプリケートするこの IAM ロールを引き受けます。

これらの最小要件に加えて、以下のオプションを選択できます。

- レプリカストレージクラス – デフォルトでは、Simple Storage Service (Amazon S3) はレプリケート元オブジェクトと同じストレージクラスを使用して、オブジェクトのレプリカを保存します。レプリカには別のストレージクラスを指定できます。
- レプリカの所有権 – Simple Storage Service (Amazon S3) は、オブジェクトレプリカが引き続きレプリケート元オブジェクトの所有者によって所有されていると想定します。そのため、オブジェクトをレプリケートするときに、対応するオブジェクトアクセスコントロールリスト (ACL) または S3 オブジェクトの所有権の設定もレプリケートします。レプリケート元とレプリケート先のバケットが異なる AWS アカウントによって所有されている場合、レプリケート先バケットを所有する AWS アカウントにレプリカの所有者を変更するようにレプリケーションを設定できます。

REST API、AWS SDK、AWS Command Line Interface (AWS CLI)、または Simple Storage Service (Amazon S3) コンソールを使用してレプリケーションを設定できます。

Simple Storage Service (Amazon S3) は、レプリケーションルールの設定をサポートする API も提供します。詳細については、Amazon Simple Storage Service API リファレンスの次のトピックを参照してください。

- PutBucketReplication
- GetBucketReplication
- DeleteBucketReplication

トピック

- レプリケーション設定
- ライブレプリケーションのアクセス許可の設定
- ライブレプリケーションの設定例

レプリケーション設定

Amazon S3 はレプリケーション設定を XML 形式で保存します。レプリケーション設定 XML ファイルで、AWS Identity and Access Management (IAM) ロールと 1 つ以上のルールを指定します。

```
<ReplicationConfiguration>
    <Role>IAM-role-ARN</Role>
    <Rule>
        ...
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
</ReplicationConfiguration>
```

Amazon S3 はユーザーの許可なしにオブジェクトをレプリケートすることはできません。レプリケーション設定で指定した IAM ロールを使用してアクセス許可を付与します。Amazon S3 は、ユーザーに代わってオブジェクトをレプリケートするための IAM ロールを引き受けます。まずは必要なアクセス許可を IAM ロールに付与する必要があります。これらのアクセス許可の管理の詳細については、「ライブレプリケーションのアクセス許可の設定」を参照してください。

次のシナリオでは、レプリケーション設定にルールを 1 つ追加します。

- すべてのオブジェクトをレプリケートします。
- オブジェクトのサブセットをレプリケートします。ルールにフィルターを追加して、オブジェクトのサブセットを特定します。フィルターでは、ルールが適用されるオブジェクトのサブセットを特

定するために、オブジェクトキープレフィックス、タグ、またはその両方の組み合わせを指定します。フィルターは、指定した値と完全に一致するオブジェクトをターゲットにします。

オブジェクトの異なるサブセットをレプリケートする場合は、レプリケーション設定に複数のルールを追加します。各ルールでは、オブジェクトの異なるサブセットを選択するフィルターを指定します。例えば、tax/ または document/ のいずれかのキープレフィックスを持つオブジェクトをレプリケートするとします。このためには、tax/ キープレフィックスフィルターを指定するルールと、document/ キープレフィックスを指定するもう 1 つのルールの 2 つのルールを追加します。オブジェクトキーのプレフィックスの詳細については、「プレフィックスを使用してオブジェクトを整理する」を参照してください。

ここで示している各セクションで、さらに詳しく学習できます。

トピック

- 基本的なルールの設定
- オプション: フィルターの指定
- 追加のレプリケート先の設定
- レプリケーション設定の例
- 下位互換性

基本的なルールの設定

各ルールには、そのルールのステータスと優先順位を含める必要があります。また、ルールに削除マーカーをレプリケートするかどうかを指定する必要もあります。

- Status は、Enabled または Disabled の値を使用して、ルールが有効か無効かを示します。ルールが無効な場合、Amazon S3 はそのルールで指定されているアクションを実行しません。
- Priorityは、複数のレプリケーションルールが競合する際に優先するルールを示します。Simple Storage Service (Amazon S3) は、すべてのレプリケーションルールに従ってオブジェクトをレプリケートしようと試みます。ただし、同じレプリケート先バケットを持つルールが 2 つ以上ある場合は、優先度が最も高いルールに従ってオブジェクトがレプリケートされます。数値が大きいほど、優先度が高くなります。
- DeleteMarkerReplication は値 Enabled または Disabled を使用して削除マーカーをレプリケートするかどうかを示します。

レプリケート先設定では、Amazon S3 にオブジェクトをレプリケートするバケットの名前を指定する必要があります。

次の例は、V2 ルールの最小要件を示しています。Amazon S3 は、下位互換性のために、引き続き XML V1 形式をサポートしています。 を参照してください。詳細については、「下位互換性」を参照してください。

```
...
    <Rule>
        <ID>Rule-1</ID>
        <Status>Enabled-or-Disabled</Status>
        <Filter>
            <Prefix></Prefix>
        </Filter>
        <Priority>integer</Priority>
        <DeleteMarkerReplication>
           <Status>Enabled-or-Disabled</Status>
        </DeleteMarkerReplication>
        <Destination>
           <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
        </Destination>
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
...
```

他の設定オプションも指定できます。たとえば、コピー元オブジェクトのストレージクラスとは異なるクラスを、オブジェクトレプリカのストレージクラスで使用するように選択できます。

オプション: フィルターの指定

ルールが適用されるオブジェクトのサブセットを選択するには、オプションのフィルタを追加します。オブジェクトキープレフィックス、オブジェクトタグ、またはその両方の組み合わせでフィルターできます。キープレフィックスとオブジェクトタグの両方でフィルターする場合、Simple Storage Service (Amazon S3) は論理 AND 演算子を使用してフィルターを組み合わせます。つまり、このルールは、特定のキープレフィックスと特定のタグを持つオブジェクトのサブセットに適用されます。

オブジェクトのキープレフィックスに基づいたフィルタリング

オブジェクトキープレフィックスに基づくフィルタを使用してルールを指定するには、次のコードを使用します。指定できるプレフィックスは 1 つだけです。

```
<Rule>
    ...
    <Filter>
        <Prefix>key-prefix</Prefix>
    </Filter>
    ...
</Rule>
...
```

オブジェクトタグに基づいたフィルタリング

オブジェクトタグに基づくフィルタを使用してルールを指定するには、次のコードを使用します。複数のオブジェクトタグを指定できます。

```
<Rule>
    ...
    <Filter>
        <And>
            <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
            </Tag>
            <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
            </Tag>
             ...
        </And>
    </Filter>
    ...
</Rule>
...
```

キープレフィックスとオブジェクトタグを使用したフィルタリング

キープレフィックスとオブジェクトタグの組み合わせでルールフィルターを指定するには、次のコードを使用します。これらのフィルターは、<And> 親要素でラップします。Amazon S3 は、これらのフィルターを結合する論理 AND オペレーションを実行します。つまり、このルールは、特定のキープレフィックスと特定のタグの両方を持つオブジェクトのサブセットに適用されます。

```
<Rule>
    ...
    <Filter>
        <And>
            <Prefix>key-prefix</Prefix>
            <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
            </Tag>
            <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
            </Tag>
             ...
    </Filter>
    ...
</Rule>
...
```

Note

- 空の <Filter> 要素を持つルールを指定すると、ルールはバケット内にある全オブジェクトに適用されます。
- ライブレプリケーションでタグベースのレプリケーションルールを使用する場合、PutObject オペレーションで一致するレプリケーションルールタグを新しいオブジェクトにタグ付けする必要があります。そうしないと、オブジェクトはレプリケートされません。PutObject オペレーション後にオブジェクトにタグが付けされると、それらのオブジェクトもレプリケートされません。

  PutObject オペレーション後にタグ付けされたオブジェクトをレプリケートするには、S3 バッチレプリケーションを使用する必要があります。バッチレプリケーションの詳細については、「既存のオブジェクトのレプリケーション」を参照してください。

追加のレプリケート先の設定

レプリケート先設定では、Amazon S3 がオブジェクトをレプリケートするバケットを指定します。1 つのレプリケート元バケットから 1 つまたは複数のレプリケート先バケットに、オブジェクトをレプリケートするように設定できます。

```
...
<Destination>
    <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
</Destination>
...
```

次のオプションを<Destination>要素に追加できます。

トピック

- ストレージクラスを指定する
- 複数の宛先バケットを追加する
- 複数のレプリケート先バケットを持つレプリケーションルールごとに異なるパラメータを指定します。
- レプリカの所有者を変更する
- S3 レプリケーション時間コントロールの有効化
- AWS KMS を使用してサーバー側の暗号化で作成されたオブジェクトをレプリケートする

ストレージクラスを指定する

オブジェクトレプリカのストレージクラスを指定できます。デフォルトでは、Amazon S3 は、レプリケート元オブジェクトのストレージクラスを使用してオブジェクトレプリカを作成します。以下に例を示します。

```
...
<Destination>
       <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
       <StorageClass>storage-class</StorageClass>
</Destination>
...
```

複数の宛先バケットを追加する

次のように、単一のレプリケーション設定に複数のレプリケート先バケットを追加できます。

```
...
<Rule>
    <ID>Rule-1</ID>
    <Status>Enabled-or-Disabled</Status>
```

```
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled-or-Disabled</Status>
    </DeleteMarkerReplication>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
</Rule>
<Rule>
    <ID>Rule-2</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled-or-Disabled</Status>
    </DeleteMarkerReplication>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET2</Bucket>
    </Destination>
</Rule>
...
```

複数のレプリケート先バケットを持つレプリケーションルールごとに異なるパラメータを指定します。

1 つのレプリケーション設定に複数のレプリケート先バケットを追加する場合、次のように、レプリケーションルールごとに異なるパラメータを指定できます。

```
...
<Rule>
    <ID>Rule-1</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Disabled</Status>
    </DeleteMarkerReplication>
      <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
```

```
    </Destination>
</Rule>
<Rule>
    <ID>Rule-2</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled</Status>
    </DeleteMarkerReplication>
      <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
  <ReplicationTime>
    <Status>Enabled</Status>
    <Time>
      <Minutes>15</Minutes>
    </Time>
  </ReplicationTime>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET2</Bucket>
    </Destination>
</Rule>
...
```

レプリカの所有者を変更する

レプリケート元とレプリケート先のバケットが同じアカウントの所有でない場合は、レプリケート先のバケットを所有する AWS アカウント にレプリカの所有権を変更できます。そのためには、AccessControlTranslation 要素を追加します。この要素は値 Destination を取ります。

```
...
<Destination>
   <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
   <Account>destination-bucket-owner-account-id</Account>
   <AccessControlTranslation>
       <Owner>Destination</Owner>
   </AccessControlTranslation>
</Destination>
...
```

この要素をレプリケーション設定に AccessControlTranslation 要素を追加しない場合は、レプリカはレプリケート元オブジェクトを所有する AWS アカウントの所有になります。詳細については、「レプリカ所有者の変更」を参照してください。

S3 レプリケーション時間コントロールの有効化

S3 Replication Time Control (S3 RTC) はレプリケーション設定で有効にできます。S3 RTC は、ほとんどのオブジェクトを数秒でレプリケートし、オブジェクトの 99.99% を 15 分以内にレプリケートします (サービスレベルアグリーメントに基づく)。

> **Note**
>
> EventThreshold と Time で受け入れられるのは、<Minutes>15</Minutes> の値のみです。

```
...
<Destination>
  <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
  <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
  <ReplicationTime>
    <Status>Enabled</Status>
    <Time>
      <Minutes>15</Minutes>
    </Time>
  </ReplicationTime>
</Destination>
...
```

詳細については、「S3 Replication Time Control (S3 RTC) を使用してコンプライアンス要件を満たす」を参照してください。API の例については、Amazon Simple Storage Service API リファレンスの「PutBucketReplication」を参照してください。

AWS KMS を使用してサーバー側の暗号化で作成されたオブジェクトをレプリケートする

レプリケート元バケットには、AWS Key Management Service (AWS KMS) (SSE-KMS) キーを使用したサーバー側の暗号化で作成されたオブジェクトが含まれている場合があります。デフォルトでは、Amazon S3 はこれらのオブジェクトをレプリケートしません。オプションで、これらのオブジェクトをレプリケートするように Amazon S3 に指示できます。これを行うには、まず、SourceSelectionCriteria 要素を追加することで、この機能を明示的にオプトインします。次に、オブジェクトレプリカの暗号化に使用するための AWS KMS key (レプリケート先バケットの AWS リージョン 用) を入力します。次の例は、これらの要素を指定する方法を示しています。

```
...
<SourceSelectionCriteria>
  <SseKmsEncryptedObjects>
    <Status>Enabled</Status>
  </SseKmsEncryptedObjects>
</SourceSelectionCriteria>
<Destination>
  <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
  <EncryptionConfiguration>
    <ReplicaKmsKeyID>AWS KMS key ID to use for encrypting object replicas</
ReplicaKmsKeyID>
  </EncryptionConfiguration>
</Destination>
...
```

詳細については、「暗号化されたオブジェクトのレプリケート (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)」を参照してください。

レプリケーション設定の例

開始するには、必要に応じて次のレプリケーション設定例をバケットに追加します。

> **Important**
>
> レプリケーション設定をバケットに追加するには、iam:PassRole アクセス許可が必要です。このアクセス許可により、Amazon S3 レプリケーションアクセス許可を付与する IAM ロールを渡すことができます。IAM ロールを指定するには、レプリケーション設定 XML の Role 要素で使用されている Amazon リソースネーム (ARN) を指定します。詳細については、IAM ユーザーガイドの「AWS のサービス サービスにロールを渡すアクセス権限をユーザーに付与する」を参照してください。

Example 1: ルールが 1 つのレプリケーション設定

次の基本的なレプリケーション設定では 1 つのルールを指定します。このルールは、Amazon S3 が引き受けることができる IAM ロールと、オブジェクトレプリカ用のレプリケート先バケットを指定します。Enabled の Status 値は、ルールが有効であることを示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>

    <Destination><Bucket>arn:aws:s3:::example-s3-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

レプリケートするオブジェクトのサブセットを選択するために、フィルタを追加できます。次の設定では、フィルタはオブジェクトキープレフィックスを指定します。このルールは、キー名に *Tax/* というプレフィックスが付いているオブジェクトに適用されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>

    <Filter>
       <Prefix>Tax/</Prefix>
    </Filter>

    <Destination><Bucket>arn:aws:s3:::example-s3-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

Filter 要素を指定した場合、Priority 要素と DeleteMarkerReplication 要素も含める必要があります。この例では、ルールが 1 つしかないため、Priority は無関係です。

次の設定では、フィルタは 1 つのプレフィックスと 2 つのタグを指定します。このルールは、指定されたキープレフィックスとタグを持つオブジェクトのサブセットに適用されます。具体的には、キー名に *Tax/* プレフィックスがあり、2 つの指定されたオブジェクトタグがあるオブジェクトに適用されます。Priority はルールが 1 つしかないため該当しません。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>

    <Filter>
        <And>
          <Prefix>Tax/</Prefix>
          <Tag>
             <Tag>
                <Key>tagA</Key>
                <Value>valueA</Value>
             </Tag>
          </Tag>
          <Tag>
             <Tag>
                <Key>tagB</Key>
                <Value>valueB</Value>
             </Tag>
          </Tag>
       </And>

    </Filter>

    <Destination><Bucket>arn:aws:s3:::example-s3-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

オブジェクトレプリカに対し、次のようにストレージクラスを指定できます。

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Destination>
       <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
       <StorageClass>storage-class</StorageClass>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

Amazon S3 がサポートする任意のストレージクラスを指定できます。

Example 2: ルールが 2 つのレプリケーション設定

Example

次のレプリケーション設定について考えます。

- 各ルールは異なるキープレフィックスでフィルタリングするため、各ルールはオブジェクトの個別のサブセットに適用されます。例えば、Simple Storage Service (Amazon S3) は、キー名 *Tax/doc1.pdf* と *Project/project1.txt* を持つオブジェクトをレプリケートしますが、キー名 *PersonalDoc/documentA* のオブジェクトはレプリケートされません。
- ルールは 2 つの異なるオブジェクトのセットに適用されるため、ルールの優先順位は関係ありません。次の例は、ルール優先順位が適用されたときに何が起こるかを示しています。
- 2 番目のルールは、オブジェクトレプリカの S3 標準 – IA ストレージクラスを指定します。Amazon S3 は、これらのオブジェクトレプリカに対して指定されたストレージクラスを使用します。

```
<?xml version="1.0" encoding="UTF-8"?>

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
```

```
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>Tax</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
     ...
  </Rule>
 <Rule>
    <Status>Enabled</Status>
    <Priority>2</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>Project</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
     <StorageClass>STANDARD_IA</StorageClass>
    </Destination>
     ...
  </Rule>


</ReplicationConfiguration>
```

Example 3: プレフィックスが重複している 2 つのルールを使用したレプリケーション設定

この設定では、2 つのルールで、重複するキープレフィックス、*star/* および *starship/* を持つフィルターを指定します。どちらのルールも、キー名が *starship-x* のオブジェクトに適用されます。この場合、Amazon S3 はルールの優先順位を使用して適用するルールを決定します。数値が大きいほど、優先度が高くなります。

```
<ReplicationConfiguration>

  <Role>arn:aws:iam::account-id:role/role-name</Role>

  <Rule>
    <Status>Enabled</Status>
```

```
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>star</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
  </Rule>
  <Rule>
    <Status>Enabled</Status>
    <Priority>2</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>starship</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

Example 4: チュートリアル例

チュートリアルについては、「ライブレプリケーションの設定例」を参照してください。

レプリケーション設定の XML 構造の詳細については、Amazon Simple Storage Service API リファレンスの「PutBucketReplication」を参照してください。

下位互換性

レプリケーション設定 XML の最新バージョンは V2 です。XML V2 レプリケーション設定は、ルールの `Filter` 要素と S3 Replication Time Control (S3 RTC) を指定するルールを含むものです。

レプリケーション設定のバージョンを確認するには、`GetBucketReplication` API オペレーションを使用します。詳細については、「Amazon Simple Storage Service API リファレンス」の「GetBucketReplication」を参照してください。

Amazon S3 は、後方互換性のために引き続き XML V1 のレプリケーション設定をサポートしています。XML V1 のレプリケーション設定を使用していた場合は、後方互換性に影響を及ぼす以下の点を考慮してください。

- レプリケーション設定 XML V2 では、ルールに `Filter` 要素が含まれています。`Filter` 要素を使用すると、オブジェクトキープレフィックス、タグ、またはその両方に基づいてオブジェクトフィルタを指定して、ルールが適用されるオブジェクトを範囲指定できます。レプリケーション設定 XML V1 は、キープレフィックスのみに基づくフィルタリングをサポートしていました。この場合は、`Prefix` を `Rule` 要素の子要素として直接追加します。以下に例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Prefix>key-prefix</Prefix>
    <Destination><Bucket>arn:aws:s3:::example-s3-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

  Amazon S3 は、下位互換性のために引き続き V1 設定をサポートしています。

- オブジェクトのバージョン ID を指定せずにレプリケート元バケットからオブジェクトを削除すると、Amazon S3 は削除マーカーを追加します。V1 のレプリケーション設定 XML を使用した場合、Simple Storage Service (Amazon S3) はユーザーアクションから発生した削除マーカーをレプリケートします。つまり、Amazon S3 はユーザーがオブジェクトを削除した場合にのみ、削除マーカーをレプリケートします。期限切れのオブジェクトが (ライフサイクルアクションの一部として) Amazon S3 によって削除された場合、Amazon S3 は削除マーカーをレプリケートしません。

  V2 のレプリケーション設定では、タグベース以外のルールのために削除マーカーレプリケーションを有効にすることができます。詳細については、「バケット間での削除マーカーのレプリケーション」を参照してください。

ライブレプリケーションのアクセス許可の設定

ライブレプリケーションを設定する場合は、次のように必要なアクセス許可を取得します。

- Simple Storage Service (Amazon S3) は、ユーザーに代わってオブジェクトをレプリケートするための許可を必要とします。IAM ロールを作成してこれらの許可を付与し、その後、レプリケーション設定でそのロールを指定します。
- レプリケート元バケットとレプリケート先バケットが同じアカウントによって所有されていない場合、レプリケート先バケット所有者は、レプリカを保存するためのアクセス許可をレプリケート元バケット所有者に付与する必要があります。

トピック

IAM ロールの作成

デフォルトで、すべての Simple Storage Service (Amazon S3) リソース (バケット、オブジェクト、関連するサブリソース) はプライベートであり、リソース所有者のみがリソースにアクセスできます。Amazon S3 には、ソースバケットからオブジェクトを読み取って、レプリケートするアクセス許可が必要です。これらのアクセス許可を付与するには、IAM ロールを作成し、レプリケーション設定でそのロールを指定します。

このセクションでは、信頼ポリシーと最低限必要なアクセス許可ポリシーについて説明します。チュートリアル例では、IAM ロールを作成するための手順をステップバイステップで説明しています。詳細については、「ライブレプリケーションの設定例」を参照してください。

- 以下の例は、信頼ポリシーを示しています。ここでは、このロールを引き受けることができるサービスプリンシパルとして Simple Storage Service (Amazon S3) を特定します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
```

```
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

- 以下の例は、信頼ポリシーを示しています。ここでは、サービスプリンシパルとして Amazon S3 と S3 バッチオペレーションを特定します。これは、バッチレプリケーションジョブを作成する場合に便利です。詳細については、「最初のレプリケーションルールまたは新しいレプリケート先にバッチレプリケーションジョブを作成する」を参照してください。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service": [
              "s3.amazonaws.com",
              "batchoperations.s3.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

IAM ロールの詳細については、「IAM ユーザーガイド」の「IAM ロール」を参照してください。

- 以下の例は、アクセスポリシーを示しています。ここでは、ユーザーに代わってレプリケーションタスクを実行する権許可をロールに付与します。Amazon S3 がこのロールを引き受ける場合、このポリシーで指定されたアクセス許可を持つことになります。このポリシーでは、*example-s3-bucket1* はレプリケート元バケットで、*example-s3-bucket2* はレプリケート先バケットです。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
```

```
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::example-s3-bucket1"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3:::example-s3-bucket1/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket2/*"
      }
   ]
}
```

アクセスポリシーは、以下のアクションに対するアクセス許可を付与します。

- `s3:GetReplicationConfiguration` および `s3:ListBucket` — バケットでのこれらのアクションの許可により、Simple Storage Service (Amazon S3) はレプリケーション設定とリストバケットのコンテンツを取得できます。(現在の権限モデルには、`s3:ListBucket` 削除マーカーにアクセスするために許可が必要です。)
- `s3:GetObjectVersionForReplication` および `s3:GetObjectVersionAcl` – すべてのオブジェクトに付与されているこれらのアクションの許可で、Simple Storage Service (Amazon S3) はオブジェクトに関連付けられた特定のオブジェクトバージョンとアクセスコントロールリスト (ACL) を取得することができます。

- `s3:ReplicateObject` および `s3:ReplicateDelete` — *example-s3-bucket2* バケットのオブジェクトに属すこれらのアクションの許可により、Simple Storage Service (Amazon S3) はバケットにオブジェクトまたは削除マーカーをレプリケートできます。削除マーカーの詳細については、「削除オペレーションがレプリケーションに与える影響」を参照してください。

  > **Note**
  >
  > *example-s3-bucket2* バケット (レプリケート先バケット) に対する `s3:ReplicateObject` アクションの許可により、オブジェクトタグや ACL などのメタデータのレプリケーションも許可されます。したがって、`s3:ReplicateTags` アクションの許可を明示的に付与する必要はありません。

- `s3:GetObjectVersionTagging` — バケット (レプリケート元バケット) のオブジェクトに対するこのアクションのアクセス許可により、Amazon S3 はレプリケーションのためにオブジェクトタグを読み取ることができるようになります。詳細については、「タグを使用してストレージを分類する」を参照してください。Amazon S3 がこれらのアクセス許可を持っていない場合は、オブジェクトはレプリケートされますが、オブジェクトタグはレプリケートされません。

Amazon S3 アクションの一覧については、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

> **Important**
>
> IAM ロールを所有する AWS アカウントは、IAM ロールに付与するアクションの許可を持っている必要があります。
> 例えば、レプリケート元バケットに別の AWS アカウントが所有するオブジェクトが含まれていたとします。オブジェクトの所有者は、IAM ロールを所有する AWS アカウントに、オブジェクト ACL を介して必要な許可を明示的に付与する必要があります。そうでない場合、Amazon S3 はオブジェクトにアクセスできず、オブジェクトのレプリケーションは失敗します。ACL のアクセス許可については、「アクセスコントロールリスト (ACL) の概要」を参照してください。
> ここで説明されているアクセス許可は、最小のレプリケーション設定に関連しています。オプションのレプリケーション設定を追加する場合は、追加のアクセス許可を Amazon S3 に付与する必要があります。

レプリケーション元とレプリケーション先のバケットが異なる AWS アカウントによって所有されている場合の許可の付与

ソースバケットと宛先バケットが同じアカウントによって所有されていない場合、レプリケート先バケット所有者は、次のように、レプリケート元バケット所有者にレプリケーションアクションを実行する許可を付与するバケットポリシーも追加する必要があります。このポリシーでは、*example-s3-bucket2* はレプリケーション先バケットです。

> **Note**
>
> ロールの ARN 形式が異なって表示される場合があります。コンソールを使用してロールが作成された場合、ARN 形式は arn:aws:iam::*account-ID*:role/service-role/*role-name* です。AWS CLI を使用してロールが作成された場合、ARN 形式は arn:aws:iam::*account-ID*:role/*role-name* です。詳細については、「IAM ユーザーガイド」の「IAM ロール」を参照してください。

```
{
   "Version":"2012-10-17",
   "Id":"PolicyForDestinationBucket",
   "Statement":[
      {
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3:ReplicateDelete",
            "s3:ReplicateObject"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket2/*"
      },
      {
         "Sid":"Permissions on bucket",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
```

```
         "Action": [
            "s3:List*",
            "s3:GetBucketVersioning",
            "s3:PutBucketVersioning"
         ],
         "Resource":"arn:aws:s3::*example-s3-bucket2*"
      }
   ]
}
```

例については、「レプリケート元バケットとレプリケート先バケットが異なるアカウントによって所有されている場合での、レプリケーションの設定」を参照してください。

レプリケート元バケットのオブジェクトがタグ付きの場合は、以下の点に注意してください。

- レプリケート元のバケット所有者が (IAM ロールによって) オブジェクトタグをレプリケートするための s3:GetObjectVersionTagging および s3:ReplicateTags アクションのアクセス許可を Amazon S3 に付与した場合、Amazon S3 はオブジェクトと共にタグをレプリケートします。IAM ロールに関する詳細は、「IAM ロールの作成」を参照してください。
- レプリケート先のバケット所有者は、タグがレプリケートされるのを望まない場合、次のステートメントをレプリケート先のバケットポリシーに追加することで、s3:ReplicateTags アクションのアクセス許可を明示的に拒否できます。このポリシーでは、*example-s3-bucket2* はレプリケーション先バケットです。

```
...
   "Statement":[
      {
         "Effect":"Deny",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-id:role/service-role/source-account-IAM-role"
         },
         "Action":"s3:ReplicateTags",
         "Resource":"arn:aws:s3:::example-s3-bucket2/*"
      }
   ]
...
```

S3 バッチオペレーションに対するアクセス許可の付与

S3 バッチレプリケーションを使用すると、レプリケーション設定が実行される前に存在していたオブジェクト、以前にレプリケートされたオブジェクト、およびレプリケーションに失敗したオブジェクトをレプリケートできます。新しいレプリケーション設定で最初のルールを作成する場合や、AWS Management Console を介して新しいレプリケート先を既存の設定に追加する場合は、1 回限りのバッチレプリケーションジョブを作成できます。バッチオペレーションジョブを作成して、既存のレプリケーション設定でバッチレプリケーションを開始することもできます。

バッチレプリケーションの IAM ロールとポリシーの例については、「バッチレプリケーション用の IAM ポリシーの設定」を参照してください。

レプリカの所有権の変更

レプリケート元とレプリケート先のバケットが、異なる AWS アカウントによって所有されている場合、レプリケート先バケットを所有する AWS アカウントにレプリカの所有者を変更するように Amazon S3 に指示できます。所有者上書きの詳細については、「レプリカ所有者の変更」を参照してください。

レプリケートされたオブジェクトをレプリケート元バケットから受信できるようにする

レプリケートされたオブジェクトをレプリケート元バケットから AWS Management Console を通して受信できるようにするために必要なポリシーを迅速に生成できます。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. バケットリストで、レプリケート先バケットとして使用するバケットを選択します。
4. [Management (管理)] タブを選択し、[Replication rules (レプリケーションルール)] まで下にスクロールします。
5. [Actions] (アクション)で、[Receive replicated objects] (レプリケートされたオブジェクトの受信) を選択します。

   プロンプトに従って、レプリケート元バケットアカウントの AWS アカウント ID を選択し、[[Generate policies] (ポリシーの生成) を選択します。これにより、Amazon S3 バケットポリシーと KMS キーポリシーが生成されます。
6. このポリシーを既存のバケットポリシーに追加するには、[Apply settings] (設定の適用) を選択するか、または [Copy] (コピー) をクリックして、変更を手動でコピーします。

7. (オプション) AWS KMS ポリシーを AWS Key Management Service コンソール上の必要な KMS キーポリシーへコピーします。

ライブレプリケーションの設定例

以下の例は、一般的なユースケース用にライブレプリケーションを設定する方法を示しています。

> **Note**
>
> ライブレプリケーションとは、同一リージョンレプリケーション (SRR) およびクロスリージョンレプリケーション (CRR) を指します。ライブレプリケーションでは、レプリケーションを設定する前にバケットに存在していたオブジェクトはレプリケートされません。レプリケーションを設定する前に存在していたオブジェクトをレプリケートするには、オンデマンドレプリケーションを使用します。バケットを同期して既存のオブジェクトをオンデマンドでレプリケートするには、「既存のオブジェクトのレプリケーション」を参照してください。

これらの例は、Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、AWS SDK (AWS SDK for Java および AWS SDK for .NET の例を示します) を使用したレプリケーション設定の作成方法を示しています。

AWS CLI のインストールと設定方法の詳細については、AWS Command Line Interface ユーザーガイドの次のトピックを参照してください。

- AWS Command Line Interface のインストール
- AWS CLI の設定 - 少なくとも 1 つのプロファイルを設定する必要があります。クロスアカウントのシナリオを学習する場合は、2 つのプロファイルを設定します。

AWS SDK については、「AWS SDK for Java」および「AWS SDK for .NET」を参照してください。

> **Tip**
>
> ライブレプリケーションを使用してデータをレプリケートする方法の詳細な手順を説明するチュートリアルについては、「Amazon S3 レプリケーションを使用して、AWS リージョン内およびリージョン間でデータをレプリケートする」を参照してください。

トピック

- 同じアカウントが所有するレプリケート元バケットとレプリケート先バケットのレプリケーションの設定
- レプリケート元バケットとレプリケート先バケットが異なるアカウントによって所有されている場合での、レプリケーションの設定
- S3 Replication Time Control (S3 RTC) を使用してコンプライアンス要件を満たす
- 暗号化されたオブジェクトのレプリケート (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)
- Amazon S3 レプリカの変更同期によるメタデータ変更のレプリケート
- バケット間での削除マーカーのレプリケーション

同じアカウントが所有するレプリケート元バケットとレプリケート先バケットのレプリケーションの設定

レプリケーションは、同一または異なる AWS リージョンにあるバケット間でオブジェクトを自動的に非同期コピーする機能です。レプリケーションでは、新しく作成されたオブジェクトおよびオブジェクトの更新が、レプリケート元バケットからレプリケート先バケットにコピーされます。詳細については、「オブジェクトのレプリケーション」を参照してください。

レプリケーションを設定するときは、レプリケート元バケットにレプリケーションルールを追加します。レプリケーションルールにより、レプリケート元のソースバケットオブジェクトと、レプリケートされたオブジェクトが保存されるレプリケート先バケットが定義されます。ルールを作成して、バケット内のすべてのオブジェクト、または特定のキー名のプレフィックス、1 つ以上のオブジェクトタグ、あるいはその両方を持つオブジェクトのサブセットをレプリケートできます。レプリケート先のバケットはレプリケート元バケットと同じ AWS アカウントにあっても、別のアカウントにあってもかまいません。

削除するオブジェクトバージョンの ID を指定した場合、Amazon S3 はソースバケット内のそのオブジェクトバージョンを削除します。しかし、レプリケート先バケット内でその削除をレプリケートすることはありません。つまり、レプリケート先バケットから同じオブジェクトバージョンを削除しません。これは悪意のある削除からデータを保護します。

バケットにレプリケーションルールを追加すると、ルールはデフォルトで有効になるため、保存するとすぐに動作を開始します。

この例では、同じ AWS アカウントがレプリケート元とレプリケート先のバケットを所有している場合の、レプリケーションのセットアップを行います。Amazon S3 コンソールを使用する例について

は、AWS Command Line Interface (AWS CLI)、AWS SDK for Java、および AWS SDK for .NET を参照してください。

S3 コンソールの使用

ソースバケットと同じ AWS アカウント に送信先バケットがある場合にレプリケーションルールを設定するには、以下の手順を実行します。

レプリケート先バケットがレプリケート元バケットとは別のアカウントにある場合は、レプリケート先バケットにバケットポリシーを追加して、レプリケート元バケットアカウントの所有者に、レプリケート先バケットのオブジェクトをレプリケートするアクセス許可を付与する必要があります。詳細については、「レプリケーション元とレプリケーション先のバケットが異なる AWS アカウントによって所有されている場合の許可の付与」を参照してください。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、目的のバケットの名前を選択します。
4. [管理] タブをクリックし、[レプリケーションルール] までスクロールして、[レプリケーションルールを作成] を選択します。
5. [レプリケーションルールの設定] セクションの、[レプリケーションルール名] で、後でルールを識別しやすいようにルールの名前を入力します。ルール名は必須であり、バケット内で一意である必要があります。
6. [ステータス] では、デフォルトで [有効] が選択されています。有効にされたルールは、保存するとすぐに機能し始めます。ルールを後から有効にする場合は、[無効] を選択します。
7. バケットに既存のレプリケーションルールがある場合は、ルールの優先順位を指定するよう指示されます。ルールの優先順位を指定して、複数のルールの範囲に含まれるオブジェクトによって引き起こされる競合を回避する必要があります。ルールが重複している場合、Amazon S3 はルールの優先度を使用して適用するルールを決定します。数値が大きいほど、優先度が高くなります。ルーティングの優先度の詳細については、「レプリケーション設定」をご覧ください。
8. [ソースバケット] には、レプリケーションソースを設定するための次のオプションがあります。
   - バケット全体をレプリケートするには、[Apply to all objects in the bucket] (バケット内のすべてのオブジェクトに適用) を選択します。
   - 同じプレフィックスを持つすべてのオブジェクトをレプリケートするには、[Limit the scope of this rule using one or more filters (1 つまたは複数のフィルターを使用してこのルールの適

用範囲を制限します)] を選択します。これにより、指定したプレフィックスで始まる名前を持つすべてのオブジェクトにレプリケーションが制限されます (pictures など)。[プレフィックス] ボックスにプレフィックスを入力します。

> **Note**
>
> フォルダの名前をプレフィックスとして入力する場合は、最後の文字に [/] (スラッシュ) を使用する必要があります (例: pictures/)。

- 1 つまたは複数のオブジェクトタグを持つすべてのオブジェクトをレプリケートするには、[タグを追加] を選択し、ボックスにキーと値のペアを入力します。別のタグを追加するには、この手順を繰り返します。プレフィックスとタグを組み合わせることができます。オブジェクトタグの詳細については、タグを使用してストレージを分類するを参照してください。

新しいレプリケーション設定 XML スキーマでは、プレフィックスとタグのフィルタリング、およびルールの優先順位付けがサポートされています。新しいスキーマの詳細については、下位互換性を参照してください。ユーザーインターフェイスの背後で機能する Amazon S3 API で使用される XML の詳細については、「レプリケーション設定」を参照してください。新しいスキーマは、レプリケーション設定 XML V2 として記述されます。

9. [レプリケート先] で、Amazon S3 がオブジェクトをレプリケートするバケットを選択します。

> **Note**
>
> レプリケート先バケットの数は、特定のパーティション内の AWS リージョンの数に制限されます。パーティションは、リージョンのグループです。AWS には、現在 aws (標準リージョン)、aws-cn (中国リージョン)、および aws-us-gov (AWS GovCloud (US) リージョン) の 3 つのパーティションがあります。レプリケート先バケット制限の増加をリクエストするには、サービスのクォータを使用できます。

- アカウントのバケットにレプリケートするには、[このアカウントのバケットを選択] をクリックし、レプリケート先バケット名を入力または参照します。
- 別の AWS アカウント のバケットにレプリケートするには、[別のアカウントのバケットを指定] を選択し、レプリケート先バケットのアカウント ID とバケット名を入力します。

  レプリケート先バケットがレプリケート元バケットとは別のアカウントにある場合は、レプリケート先バケットにバケットポリシーを追加して、レプリケート元バケットアカウントの所有

者にオブジェクトをレプリケートするアクセス許可を付与する必要があります。詳細については、「レプリケーション元とレプリケーション先のバケットが異なる AWS アカウントによって所有されている場合の許可の付与」を参照してください。

必要に応じて、レプリケート先バケット内の新しいオブジェクトの所有を標準化するために、[オブジェクト所有者を送信先バケット所有者に変更] を選択します。このオプションの詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

> **Note**
>
> レプリケート先バケットでバージョニングが有効になっていない場合は、[バージョニングを有効化] ボタンを含む警告が表示されます。バケットでバージョニングを有効にするには、このボタンを選択します。

10. Amazon S3 がユーザーのためにオブジェクトをレプリケートできる AWS Identity and Access Management (IAM) ロールを設定します。

    IAM ロールを設定するには、[IAM ロール] セクションで [IAM ロール] ドロップダウンリストから次のいずれかを選択します。

    - [Create new role (新しいロールの作成)] を選択して、Amazon S3 で新しい IAM ロールが自動的に作成されるようにすることを強くお勧めします。ルールを保存すると、選択したレプリケート元バケットとレプリケート先バケットに一致する IAM ロールに対して新しいポリシーが生成されます。
    - 既存の IAM ロールの使用も選択できます。その場合は、レプリケーションに必要なアクセス許可を Amazon S3 に付与するロールを選択する必要があります。このロールがレプリケーションルールに従うための十分なアクセス許可を Amazon S3 に付与しない場合、レプリケーションは失敗します。

> **Important**
>
> レプリケーションルールをバケットに追加する場合は、Amazon S3 にレプリケーションアクセス許可を付与する IAM ロールを渡すことができる `iam:PassRole` アクセス許可が必要です。詳細については、IAM ユーザーガイドの「AWS のサービス のサービスにロールを渡すアクセス権限をユーザーに付与する」を参照してください。

11. AWS Key Management Service (AWS KMS) キーを使用したサーバー側の暗号化 (SSE-KMS) で暗号化されたソースバケット内のオブジェクトをレプリケートするには、[暗号化] で [AWS KMS で暗号化されたオブジェクトをレプリケート] を選択します。送信先オブジェクト暗号化用の AWS KMS キー には、レプリケーションでの使用を許可しているソースキーが示されています。デフォルトでは、すべてのソース KMS キーが含まれます。KMS キーの選択を絞り込むために、エイリアスまたはキー ID を選択できます。

    選択されていない AWS KMS keys で暗号化されたオブジェクトはレプリケートされません。KMS キーまたは KMS キーのグループが選択されていますが、必要に応じて KMS キーを選択できます。AWS KMS をレプリケーションで使用する方法については、暗号化されたオブジェクトのレプリケート (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS) を参照してください。

    > **Important**
    >
    > AWS KMS で暗号化されたオブジェクトをレプリケートすると、AWS KMS リクエストレートは、ソースリージョンでは倍になり、送信先リージョンでは同じ量だけ増加します。AWS KMS に対するこれらの呼び出しレートの増加は、レプリケーションの送信先リージョンに対して定義した KMS キーを使用してデータが再暗号化された方法によるものです。AWS KMS では、リージョンごとに呼び出しアカウントあたりのリクエストレートが制限されています。制限のデフォルト値については、AWS Key Management Service デベロッパーガイドの「AWS KMS の制限 - 1 秒あたりのリクエスト数: 可変」を参照してください。
    >
    > レプリケーションにおける現在の Amazon S3 PUT オブジェクトリクエストレートが、アカウントでデフォルトで設定されている AWS KMS レート制限の半分を超えている場合は、AWS KMS リクエストレート制限の引き上げをリクエストすることをお勧めします。引き上げをリクエストするには、AWS Support センターのお問い合わせでケースを作成します。例えば、現在の PUT オブジェクトのリクエストレートが 1 秒あたり 1,000 リクエストであり、オブジェクトの暗号化に AWS KMS を使用しているとします。この場合、レプリケート元リージョンとレプリケート先リージョン (リージョンが異なる場合) の両方で AWS KMS レート制限を毎秒 2,500 リクエストに引き上げるよう AWS Support に依頼することをお勧めします。これにより、AWS KMS によるスロットリングがなくなります。
    >
    > レプリケート元バケットの PUT オブジェクトのリクエストレートを確認するには、Amazon S3 の Amazon CloudWatch リクエストメトリクスの `PutRequests` を確認します。CloudWatch メトリクスの表示方法については、「S3 コンソールの使用」を参照してください

AWS KMS で暗号化されたオブジェクトをレプリケートを選択した場合は、以下の操作を実行します。

- [送信先オブジェクトを暗号化するための AWS KMS key] で、次のいずれかの方法で KMS キーを指定します。
  - 使用可能な KMS キーのリストから選択するには、[AWS KMS keys から選択する] を選択し、使用可能なキーのリストから自分の KMS キーを選択します。

    AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がこのリストに表示されます。カスタマーマネージドキーの詳細については、AWS Key Management Service デベロッパーガイドの「カスタマーキーと AWS キー」を参照してください。
  - KMS キー Amazon リソースネーム (ARN) を入力するには、[AWS KMS key ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。これにより、レプリケート先バケットのレプリカが暗号化されます。[IAM コンソール]の [暗号化キー] で、KMS キーの ARN を検索できます。
  - AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

    AWS KMS key の作成の詳細については、「AWS Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。

> **Important**
>
> バケットと同じ AWS リージョン で有効になっている KMS キーのみを使用できます。[KMS キーから選択する] を選択する場合、S3 コンソールにはリージョンごとに 100 個の KMS キーしか表示されません。同じリージョンに 100 個以上の KMS キーがある場合、S3 コンソールには最初の 100 個の KMS キーしか表示されません。コンソールに表示されていない KMS キーを使用するには、[AWS KMS key ARN を入力] を選択し、KMS キー ARN を入力します。
> Amazon S3 でサーバー側の暗号化に AWS KMS key を使用する場合は、対称暗号化 KMS キーを選択する必要があります。Amazon S3 では、対称暗号化 KMS キーのみがサポートされ、非対称暗号化 KMS キーはサポートされません。詳細については、AWS Key Management Service デベロッパーガイドの「Identifying

symmetric and asymmetric KMS keys」(対称および非対称 KMS キーの識別) を参照してください。

AWS KMS key の作成の詳細については、「AWS Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。Amazon S3 での AWS KMS の使用に関する詳細は、「AWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用」を参照してください。

12. [レプリケート先ストレージクラス] で、データをレプリケート先の特定のストレージクラスにレプリケートする場合は、[レプリケートされたオブジェクトのストレージクラスを変更] を選択します。次に、レプリケート先のレプリケートされたオブジェクトに使用するストレージクラスを選択します。このオプションを選択しない場合、レプリケートされたオブジェクトのストレージクラスは元のオブジェクトのクラスと同じになります。
13. 追加のレプリケーションオプションを設定する際には、次の追加オプションから選択できます。

    - レプリケーション設定で S3 Replication Time Control (S3 RTC) を有効にする場合は、[レプリケーション時間のコントロール (RTC)] を選択します。このオプションの詳細については、「S3 Replication Time Control (S3 RTC) を使用してコンプライアンス要件を満たす」を参照してください。
    - レプリケーション設定で S3 レプリケーションメトリクスを有効にするには、[Replication metrics and events] (レプリケーションメトリクスとイベント) を選択します。詳細については、「レプリケーションメトリクスと S3 イベント通知による、進捗状況のモニタリング」を参照してください。
    - レプリケーション設定で削除マーカーレプリケーションを有効にする場合は、[削除マーカーのレプリケーション] を選択します。詳細については、「バケット間での削除マーカーのレプリケーション」を参照してください。
    - レプリケーション設定で Amazon S3 レプリカ変更の同期を有効にする場合は、[レプリカ変更の同期] を選択します。詳細については、「Amazon S3 レプリカの変更同期によるメタデータ変更のレプリケート」を参照してください。

    > **Note**
    >
    > S3 RTC または S3 レプリケーションメトリクスを使用する場合は、追加料金が適用されます。

14. 終了するには、[Save (保存)] を選択します。

15. ルールを保存したら、ルールを選択して [Edit rule (ルールの編集)] を選択することで、ルールを編集、有効化、無効化、または削除できます。

AWS CLI の使用

レプリケート元バケットとレプリケート先バケットが同じ AWS アカウント によって所有されている場合に、レプリケーションの設定に AWS CLI を使用するには

- レプリケート元バケットとレプリケート先バケットを作成する
- バケットでのバージョニングを有効化する
- オブジェクトをレプリケートするための Simple Storage Service (Amazon S3) 許可を付与する IAM ロールを作成する
- レプリケート元バケットにレプリケーション設定を追加する

設定を確認するには、テストします。

レプリケート元バケットとレプリケート先バケットが同じ AWS アカウントによって所有されている場合にレプリケーションをセットアップするには

1. AWS CLI の認証情報プロファイルを設定します。この例では、プロファイル名 `acctA` を使用します。認証情報プロファイルの設定については、AWS Command Line Interface ユーザーガイドの「名前付きプロファイル」を参照してください。

   > **Important**
   >
   > この演習に使用するプロファイルは、必要なアクセス許可を持っている必要があります。たとえば、レプリケーション設定で、Amazon S3 が引き受けることができる IAM ロールを指定します。使用するプロファイルに `iam:PassRole` アクセス権限がある場合のみ実行できます。詳細については、IAM ユーザーガイドの「AWS サービスにロールを渡すアクセス権限をユーザーに付与する」を参照してください。管理者の認証情報を使用して名前付きプロファイルを作成すると、すべてのタスクを実行できるようになります。

2. *`source`* バケットを作成してバージョニングを有効にします。次のコードは、米国東部 (バージニア北部) (us-east-1) リージョンに *`source`* バケットを作成します。

```
aws s3api create-bucket \
```

```
--bucket source \
--region us-east-1 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket source \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. *destination* バケットを作成してバージョニングを有効にします。次のコードは、米国西部 (オレゴン) (us-west-2) リージョンに *destination* バケットを作成します。

> **Note**
>
> レプリケーション元とレプリケーション先バケットの両方が同じ AWS アカウントにある場合、レプリケーション設定をセットアップするには、同じプロファイルを使用します。この例では acctA を使用します。異なる AWS アカウントによってバケットが所有されている場合、レプリケーション設定をテストするには、それぞれに異なるプロファイルを指定します。この例では、レプリケーション先バケットに acctB プロファイルを使用します。

```
aws s3api create-bucket \
--bucket destination \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket destination \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. IAM ロールを作成します。*#######*バケットに後で追加するレプリケーション設定でこのロールを指定します。Amazon S3 は、ユーザーに代わってオブジェクトをレプリケートするこのロールを引き受けます。IAM ロールは 2 つのステップで作成します。

   - ロールを作成します。

- アクセス権限ポリシーをロールにアタッチします。

a. IAM ロールを作成します。

i. 次の信頼ポリシーをコピーして、ローカルコンピュータの現在のディレクトリにある s3-role-trust-policy.json という名前のファイルに保存します。このポリシーは、ロールを引き受けるアクセス許可を Amazon S3 サービスプリンシパルに付与します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

ii. 次のコマンドを実行して、ロールを作成します。

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
--profile acctA
```

b. アクセス権限ポリシーをロールにアタッチします。

i. 次のアクセス権限ポリシーをコピーして、ローカルコンピュータの現在のディレクトリにある s3-role-permissions-policy.json という名前のファイルに保存します。このポリシーは、さまざまな Amazon S3 バケットとオブジェクトアクションに対するアクセス許可を付与します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
```

```
            "Action":[
               "s3:GetObjectVersionForReplication",
               "s3:GetObjectVersionAcl",
               "s3:GetObjectVersionTagging"
            ],
            "Resource":[
               "arn:aws:s3:::source-bucket/*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:ListBucket",
               "s3:GetReplicationConfiguration"
            ],
            "Resource":[
               "arn:aws:s3:::source-bucket"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:ReplicateObject",
               "s3:ReplicateDelete",
               "s3:ReplicateTags"
            ],
            "Resource":"arn:aws:s3:::destination-bucket/*"
         }
      ]
}
```

ii. ポリシーを作成してロールにアタッチするには、次のコマンドを実行します。

```
$ aws iam put-role-policy \
--role-name replicationRole \
--policy-document file://s3-role-permissions-policy.json \
--policy-name replicationRolePolicy \
--profile acctA
```

5. *source* バケットにレプリケーション設定を追加します。

a. Amazon S3 API は XML でレプリケーション設定をする必要がありますが、AWS CLI はレプリケーション設定を JSON で指定する必要があります。以下の JSON を、コンピュータのローカルディレクトリの `replication.json` というファイルに保存します。

```
{
  "Role": "IAM-role-ARN",
  "Rules": [
    {
      "Status": "Enabled",
      "Priority": 1,
      "DeleteMarkerReplication": { "Status": "Disabled" },
      "Filter" : { "Prefix": "Tax"},
      "Destination": {
        "Bucket": "arn:aws:s3:::destination-bucket"
      }
    }
  ]
}
```

b. *destination-bucket* と *IAM-role-ARN* の値を指定して JSON を更新します。変更を保存します。

c. 次のコマンドを実行して、レプリケート元バケットにレプリケーション設定を追加します。必ず *source* バケット名を指定してください。

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket source \
--profile acctA
```

レプリケーション設定を取得するには、`get-bucket-replication` コマンドを使用します。

```
$ aws s3api get-bucket-replication \
--bucket source \
--profile acctA
```

6. Amazon S3 コンソールでセットアップをテストします。

a. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

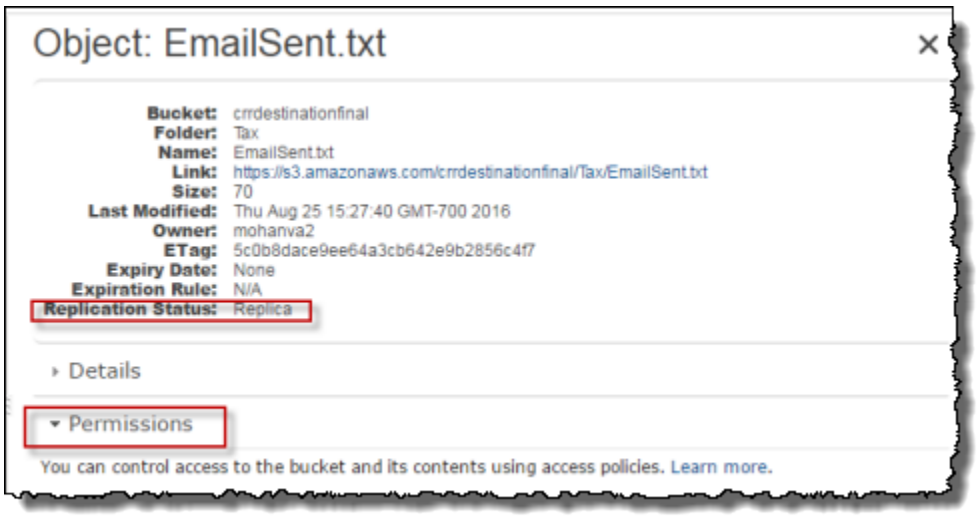
b. *source* バケットに Tax という名前のフォルダを作成します。

c. *source* バケット内の Tax フォルダにサンプルオブジェクトを追加します。

> **Note**
>
> Amazon S3 がオブジェクトをレプリケートするのにかかる時間は、オブジェクトのサイズによって異なります。レプリケーションのステータスを確認する方法については、「レプリケーションステータス情報の取得」を参照してください。

destination バケットで、以下の点を確認します。

- Amazon S3 がオブジェクトをレプリケートしたこと。
- オブジェクトの [プロパティ] で、[レプリケーションステータス] が Replica (これをレプリカオブジェクトとして識別する) に設定されていること。
- オブジェクトの [プロパティ] で、アクセス権限セクションに何もアクセス権限が表示されていないこと。これは、レプリカがまだ *source* バケット所有者によって所有されており、*destination* バケット所有者にオブジェクトレプリカの許可がないことを意味します。オプションの設定を追加して、レプリカの所有権を変更するよう Amazon S3 に指示することができます。例については、「レプリカの所有者を変更する方法」を参照してください。



AWS SDK の使用

以下のコード例を使用して、AWS SDK for Java および AWS SDK for .NET で、レプリケーション設定をバケットにそれぞれ追加します。

Java

以下の例では、まずバケットにレプリケーション設定を追加した後、その設定を取得して確認します。作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import
 com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateRoleRequest;
import com.amazonaws.services.identitymanagement.model.PutRolePolicyRequest;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.BucketReplicationConfiguration;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.CreateBucketRequest;
import com.amazonaws.services.s3.model.DeleteMarkerReplication;
import com.amazonaws.services.s3.model.DeleteMarkerReplicationStatus;
import com.amazonaws.services.s3.model.ReplicationDestinationConfig;
import com.amazonaws.services.s3.model.ReplicationRule;
import com.amazonaws.services.s3.model.ReplicationRuleStatus;
import com.amazonaws.services.s3.model.SetBucketVersioningConfigurationRequest;
import com.amazonaws.services.s3.model.StorageClass;
import com.amazonaws.services.s3.model.replication.ReplicationFilter;
import com.amazonaws.services.s3.model.replication.ReplicationFilterPredicate;
import com.amazonaws.services.s3.model.replication.ReplicationPrefixPredicate;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

```
public class CrossRegionReplication {

        public static void main(String[] args) throws IOException {
                Regions clientRegion = Regions.DEFAULT_REGION;
                String accountId = "*** Account ID ***";
                String roleName = "*** Role name ***";
                String sourceBucketName = "*** Source bucket name ***";
                String destBucketName = "*** Destination bucket name ***";
                String prefix = "Tax/";

                String roleARN = String.format("arn:aws:iam::%s:%s", accountId,
 roleName);
                String destinationBucketARN = "arn:aws:s3:::" + destBucketName;

                AmazonS3 s3Client = AmazonS3Client.builder()
                                .withCredentials(new ProfileCredentialsProvider())
                                .withRegion(clientRegion)
                                .build();

                createBucket(s3Client, clientRegion, sourceBucketName);
                createBucket(s3Client, clientRegion, destBucketName);
                assignRole(roleName, clientRegion, sourceBucketName,
 destBucketName);

                try {

                        // Create the replication rule.
                        List<ReplicationFilterPredicate> andOperands = new
 ArrayList<ReplicationFilterPredicate>();
                        andOperands.add(new ReplicationPrefixPredicate(prefix));

                        Map<String, ReplicationRule> replicationRules = new
 HashMap<String, ReplicationRule>();
                        replicationRules.put("ReplicationRule1",
                                        new ReplicationRule()
                                                        .withPriority(0)

 .withStatus(ReplicationRuleStatus.Enabled)

 .withDeleteMarkerReplication(
                                                                        new

 DeleteMarkerReplication().withStatus(

    DeleteMarkerReplicationStatus.DISABLED))
```

```
                                                    .withFilter(new
ReplicationFilter().withPredicate(
                                                            new
ReplicationPrefixPredicate(prefix)))
                                                    .withDestinationConfig(new
ReplicationDestinationConfig()

.withBucketARN(destinationBucketARN)

.withStorageClass(StorageClass.Standard)));

                        // Save the replication rule to the source bucket.
                        s3Client.setBucketReplicationConfiguration(sourceBucketName,
                                        new BucketReplicationConfiguration()
                                                    .withRoleARN(roleARN)

.withRules(replicationRules));

                        // Retrieve the replication configuration and verify that
the configuration
                        // matches the rule we just set.
                        BucketReplicationConfiguration replicationConfig = s3Client

.getBucketReplicationConfiguration(sourceBucketName);
                        ReplicationRule rule =
replicationConfig.getRule("ReplicationRule1");
                        System.out.println("Retrieved destination bucket ARN: "
                                        +
rule.getDestinationConfig().getBucketARN());
                        System.out.println("Retrieved priority: " +
rule.getPriority());
                        System.out.println("Retrieved source-bucket replication rule
status: " + rule.getStatus());
                } catch (AmazonServiceException e) {
                        // The call was transmitted successfully, but Amazon S3
couldn't process
                        // it, so it returned an error response.
                        e.printStackTrace();
                } catch (SdkClientException e) {
                        // Amazon S3 couldn't be contacted for a response, or the
client
                        // couldn't parse the response from Amazon S3.
                        e.printStackTrace();
                }
```

```
        }

        private static void createBucket(AmazonS3 s3Client, Regions region, String
 bucketName) {
                CreateBucketRequest request = new CreateBucketRequest(bucketName,
 region.getName());
                s3Client.createBucket(request);
                BucketVersioningConfiguration configuration = new
 BucketVersioningConfiguration()
                                .withStatus(BucketVersioningConfiguration.ENABLED);

                SetBucketVersioningConfigurationRequest enableVersioningRequest =
 new SetBucketVersioningConfigurationRequest(
                                bucketName, configuration);
                s3Client.setBucketVersioningConfiguration(enableVersioningRequest);

        }

        private static void assignRole(String roleName, Regions region, String
 sourceBucket, String destinationBucket) {
                AmazonIdentityManagement iamClient =
 AmazonIdentityManagementClientBuilder.standard()
                                .withRegion(region)
                                .withCredentials(new ProfileCredentialsProvider())
                                .build();
                StringBuilder trustPolicy = new StringBuilder();
                trustPolicy.append("{\\r\\n   ");
                trustPolicy.append("\\\"Version\\\":\\\"2012-10-17\\\",\\r\\n   ");
                trustPolicy.append("\\\"Statement\\\":[\\r\\n      {\\r\\n
 ");
                trustPolicy.append("\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\
\"Principal\\\":{\\r\\n            ");
                trustPolicy.append("\\\"Service\\\":\\\"s3.amazonaws.com\\\"\\r\\n
       },\\r\\n         ");
                trustPolicy.append("\\\"Action\\\":\\\"sts:AssumeRole\\\"\\r\\n
  }\\r\\n   ]\\r\\n}");

                CreateRoleRequest createRoleRequest = new CreateRoleRequest()
                                .withRoleName(roleName)

 .withAssumeRolePolicyDocument(trustPolicy.toString());

                iamClient.createRole(createRoleRequest);
```

```
                StringBuilder permissionPolicy = new StringBuilder();
                permissionPolicy.append(
                                "{\\r\\n   \\\"Version\\\":\\\"2012-10-17\\\",\\r\\n
   \\\"Statement\\\":[\\r\\n      {\\r\\n         ");
                permissionPolicy.append(
                                "\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\
\"Action\\\":[\\r\\n             ");
                permissionPolicy.append("\\\"s3:GetObjectVersionForReplication\\\",\
\r\\n            ");
                permissionPolicy.append(
                                "\\\"s3:GetObjectVersionAcl\\\"\\r\\n         ],\\r\
\n         \\\"Resource\\\":[\\r\\n            ");
                permissionPolicy.append("\\\"arn:aws:s3:::");
                permissionPolicy.append(sourceBucket);
                permissionPolicy.append("/*\\\"\\r\\n         ]\\r\\n      },\\r\\n
     {\\r\\n         ");
                permissionPolicy.append(
                                "\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\
\"Action\\\":[\\r\\n            ");
                permissionPolicy.append(
                                "\\\"s3:ListBucket\\\",\\r\\n            \\
\"s3:GetReplicationConfiguration\\\"\\r\\n         ");
                permissionPolicy.append("],\\r\\n         \\\"Resource\\\":[\\r\\n
          \\\"arn:aws:s3:::");
                permissionPolicy.append(sourceBucket);
                permissionPolicy.append("\\r\\n         ");
                permissionPolicy
                                .append("]\\r\\n      },\\r\\n      {\\r\\n
 \\\"Effect\\\":\\\"Allow\\\",\\r\\n         ");
                permissionPolicy.append(
                                "\\\"Action\\\":[\\r\\n            \\
\"s3:ReplicateObject\\\",\\r\\n            ");
                permissionPolicy
                                .append("\\\"s3:ReplicateDelete\\\",\\r\\n
  \\\"s3:ReplicateTags\\\",\\r\\n            ");
                permissionPolicy.append("\\\"s3:GetObjectVersionTagging\\\"\\r\\n\\r
\\n         ],\\r\\n         ");
                permissionPolicy.append("\\\"Resource\\\":\\\"arn:aws:s3:::");
                permissionPolicy.append(destinationBucket);
                permissionPolicy.append("/*\\\"\\r\\n      }\\r\\n   ]\\r\\n}");

                PutRolePolicyRequest putRolePolicyRequest = new
 PutRolePolicyRequest()
                                .withRoleName(roleName)
```

```
                                .withPolicyDocument(permissionPolicy.toString())
                                .withPolicyName("crrRolePolicy");

                iamClient.putRolePolicy(putRolePolicyRequest);

        }
}
```

C#

以下の AWS SDK for .NET のコード例では、バケットにレプリケーション設定を追加してから、その設定を取得します。このコードを使用するには、バケットの名前と IAM ロールの Amazon リソースネーム (ARN) を入力します。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CrossRegionReplicationTest
    {
        private const string sourceBucket = "*** source bucket ***";
        // Bucket ARN example - arn:aws:s3:::destinationbucket
        private const string destinationBucketArn = "*** destination bucket ARN
 ***";
        private const string roleArn = "*** IAM Role ARN ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint sourceBucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(sourceBucketRegion);
            EnableReplicationAsync().Wait();
        }
        static async Task EnableReplicationAsync()
        {
```

```
            try
            {
                ReplicationConfiguration replConfig = new ReplicationConfiguration
                {
                    Role = roleArn,
                    Rules =
                        {
                            new ReplicationRule
                            {
                                Prefix = "Tax",
                                Status = ReplicationRuleStatus.Enabled,
                                Destination = new ReplicationDestination
                                {
                                    BucketArn = destinationBucketArn
                                }
                            }
                        }
                };

                PutBucketReplicationRequest putRequest = new
PutBucketReplicationRequest
                {
                    BucketName = sourceBucket,
                    Configuration = replConfig
                };

                PutBucketReplicationResponse putResponse = await
s3Client.PutBucketReplicationAsync(putRequest);

                // Verify configuration by retrieving it.
                await RetrieveReplicationConfigurationAsync(s3Client);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }
```

```
        private static async Task RetrieveReplicationConfigurationAsync(IAmazonS3
 client)
        {
            // Retrieve the configuration.
            GetBucketReplicationRequest getRequest = new GetBucketReplicationRequest
            {
                BucketName = sourceBucket
            };
            GetBucketReplicationResponse getResponse = await
 client.GetBucketReplicationAsync(getRequest);
            // Print.
            Console.WriteLine("Printing replication configuration information...");
            Console.WriteLine("Role ARN: {0}", getResponse.Configuration.Role);
            foreach (var rule in getResponse.Configuration.Rules)
            {
                Console.WriteLine("ID: {0}", rule.Id);
                Console.WriteLine("Prefix: {0}", rule.Prefix);
                Console.WriteLine("Status: {0}", rule.Status);
            }
        }
    }
}
```

レプリケート元バケットとレプリケート先バケットが異なるアカウントによって所有されている場合での、レプリケーションの設定

#######バケットと#######バケットが異なる AWS アカウントによって所有されている場合のレプリケーションの設定は、両方のバケットが同じアカウントによって所有されている場合のレプリケーションの設定と似ています。唯一の違いとして、#######バケットの所有者は、#######バケットの所有者に、バケットポリシーを追加してオブジェクトをレプリケートするアクセス権限を与える必要があります。

クロスアカウントシナリオで AWS Key Management Service でのサーバー側の暗号化を使用したレプリケーションの設定の詳細については、「クロスアカウントシナリオに対する追加のアクセス許可の付与」を参照してください。

レプリケート元バケットとレプリケート先バケットが異なる AWS アカウントによって所有されている場合にレプリケーションを設定するには

1. この例では、#########バケットと#########バケットを 2 つの異なる AWS アカウントで作成します。AWS CLI には 2 つの認証情報プロファイルを設定する必要があります (この例で

は、プロファイル名に acctA と acctB を使用します)。認証情報プロファイルの設定については、AWS Command Line Interface ユーザーガイドの名前付きプロファイルを参照してください。

2. 「同じアカウントでのバケットの設定」の手順に従って、以下の変更を加えます。

   - *#######*バケットアクティビティ (*#######*バケットの作成、バージョニングの有効化、および IAM ロールの作成) に関連するすべての AWS CLI コマンドには、acctA プロファイルを使用します。acctB プロファイルを使用して*#######*バケットを作成します。
   - この例で作成した*#######*バケット、および*#######*バケットがアクセス権限ポリシーで指定されていることを確認してください。

3. コンソールで*#######*バケットに次のバケットポリシーを追加して、*#######*バケットの所有者がオブジェクトをレプリケートできるようにします。必ず、*#######*バケット所有者の AWS アカウント ID と*#######*バケット名を指定してポリシーを編集してください。

> **Note**
>
> 次の例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。*DOC-EXAMPLE-BUCKET* を送信先のバケットの名前に置き換えます。*source-bucket-acct-ID:role/Service-role/source-acct-IAM-Role* を、このレプリケーション設定に使用しているロールに置き換えます。
> IAM サービスロールを手動で作成した場合は、次のポリシー例に示すように、ロールパスを role/service-role/ として設定します。詳細については、「IAM ユーザーガイド」の「IAM ARN」を参照してください。

```
{
   "Version":"2012-10-17",
   "Id":"",
   "Statement":[
      {
         "Sid":"Set-permissions-for-objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-acct-ID:role/service-role/source-
acct-IAM-role"
         },
         "Action":["s3:ReplicateObject", "s3:ReplicateDelete"],
         "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
```

```
      },
      {
         "Sid":"Set permissions on bucket",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-acct-ID:role/service-role/source-
acct-IAM-role"
         },
         "Action":["s3:GetBucketVersioning", "s3:PutBucketVersioning"],
         "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET"
      }
   ]
}
```

バケットを選択してバケットポリシーを追加します。手順については、Amazon S3 コンソールを使用したバケットポリシーの追加 を参照してください。

レプリケーションでは、デフォルトで、レプリケート元オブジェクトの所有者もレプリカを所有しています。レプリケート元とレプリケート先のバケットが、異なる AWS アカウントによって所有されている場合、レプリケート先バケットを所有する AWS アカウントにレプリカの所有権を変更するためのオプション設定を追加できます。これには、ObjectOwnerOverrideToBucketOwner 許可の付与が含まれます。詳細については、「レプリカ所有者の変更」を参照してください。

レプリカ所有者の変更

レプリケーションでは、デフォルトで、レプリケート元オブジェクトの所有者もレプリカを所有しています。レプリケート元とレプリケート先のバケットが異なる AWS アカウント によって所有され、レプリケート先バケットを所有する AWS アカウント にレプリカの所有権を変更したい場合、オプション構成設定を追加して、レプリカの所有権を送付先バケットを所有する AWS アカウントに変更することができます。たとえば、オブジェクトのレプリカへのアクセスを制限するために、これを行うことがあります。これは、レプリケーション設定の所有者オーバーライドオプションと呼ばれます。所有者上書きオプションの詳細については、「レプリケーション設定への所有者オーバーライドオプションの追加」を参照してください。レプリケーション設定の構成については、「オブジェクトのレプリケーション」を参照してください。

所有者オーバーライドを設定するには、以下を実行します。

- レプリカの所有権を変更するよう Amazon S3 に指示するには、レプリケーション設定に所有者オーバーライドオプションを追加します。
- レプリカの所有権を変更するアクセス許可を Amazon S3 に付与します。

- レプリケート先バケットポリシーにアクセス許可を追加して、レプリカの所有権を変更できるようにします。これにより、レプリケート先バケットの所有者がオブジェクトレプリカの所有権を受け入れることができるようになります。

詳細については、「レプリケーション設定への所有者オーバーライドオプションの追加」を参照してください。ステップバイステップの手順を含む実例については、「レプリカの所有者を変更する方法」を参照してください。

オブジェクトの所有権のためのバケット所有者の強制設定

Simple Storage Service (Amazon S3) レプリケーションを使用し、レプリケート元バケットとレプリケート先バケットが異なった AWS アカウント によって所有される場合、バケットを使用する場合、レプリケート先バケットのバケット所有者は ACL を無効にして (「オブジェクト所有権」に対するバケット所有者の強制設定)、レプリカの所有権を送信先バケットを所有する AWS アカウントに変更することができます。この設定は、`s3:ObjectOwnerOverrideToBucketOwner` の許可の必要なく、既存の所有者のオーバーライドの動作を模倣します。つまり、バケット所有者の強制設定を使用してレプリケート先バケットにレプリケートされるすべてのオブジェクトは、レプリケート先バケット所有者によって所有されることを意味しています。オブジェクト所有権については、オブジェクトの所有権の制御とバケットの ACL の無効化。 を参照してください。

レプリケーション設定への所有者オーバーライドオプションの追加

> **Warning**
>
> 所有者オーバーライドオプションを追加するのは、レプリケーション元バケットとレプリケーション先バケットが異なる AWS アカウントによって所有されている場合のみです。Amazon S3 はバケットが同じアカウントによって所有されているか、異なるアカウントによって所有されているかをチェックしません。両方のバケットが同じ AWS アカウントによって所有されている場合に所有者オーバーライドを追加すると、Amazon S3 は所有者オーバーライドを適用します。レプリケート先バケットの所有者にフルアクセス権が付与され、その後の更新はソースオブジェクトアクセスコントロールリスト (ACL) にレプリケートされません。レプリカの所有者は、`PUT ACL` リクエストを持つレプリカに関連付けられた ACL を直接変更できますが、レプリケーションを通して変更することはできません。

所有者の上書きオプションを指定するには、各`Destination`要素に次を追加します。

- Amazon S3 にレプリカの所有権を変更するよう指示する `AccessControlTranslation` 要素

- レプリケート先バケット所有者の AWS アカウントを指定する Account 要素

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    ...
    <Destination>
      ...
      <AccessControlTranslation>
           <Owner>Destination</Owner>
       </AccessControlTranslation>
      <Account>destination-bucket-owner-account-id</Account>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

次のレプリケーション設定例では、Tax キープレフィックスを持つオブジェクトをレプリケート先バケットにレプリケートし、レプリカの所有権を変更するようAmazon S3 に指示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Role>arn:aws:iam::account-id:role/role-name</Role>
   <Rule>
      <ID>Rule-1</ID>
      <Priority>1</Priority>
      <Status>Enabled</Status>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::destination-bucket</Bucket>
         <Account>destination-bucket-owner-account-id</Account>
         <AccessControlTranslation>
            <Owner>Destination</Owner>
         </AccessControlTranslation>
      </Destination>
   </Rule>
</ReplicationConfiguration>
```

レプリカの所有権を変更するための Amazon S3 アクセス許可の付与

IAM ロールに関連付けられているアクセス許可ポリシーに `s3:ObjectOwnerOverrideToBucketOwner` アクションのアクセス許可を追加することで、Amazon S3 にレプリカの所有権を変更するアクセス許可を付与します。これは、Amazon S3 がユーザーに代わってオブジェクトを引き受けてレプリケートすることを可能にする、レプリケーション設定で指定した IAM ロールです。

```
...
{
    "Effect":"Allow",
         "Action":[
       "s3:ObjectOwnerOverrideToBucketOwner"
    ],
    "Resource":"arn:aws:s3:::destination-bucket/*"
}
...
```

レプリケート先バケットポリシーへのレプリカの所有権を変更するアクセス許可の追加

レプリケート先バケットの所有者は、レプリケート元バケットの所有者に、レプリカの所有権を変更するためのアクセス権限を付与する必要があります。レプリケート先バケットの所有者は、レプリケート元バケットの所有者に、`s3:ObjectOwnerOverrideToBucketOwner` アクションのためのアクセス権限を付与します。これにより、宛先バケット所有者はオブジェクトレプリカの所有権を受け入れることができるようになります。次のバケットポリシーステートメントの例は、この操作を行う方法を示しています。

```
...
{
    "Sid":"1",
    "Effect":"Allow",
    "Principal":{"AWS":"source-bucket-account-id"},
    "Action":["s3:ObjectOwnerOverrideToBucketOwner"],
    "Resource":"arn:aws:s3:::destination-bucket/*"
}
...
```

追加の考慮事項

所有権オーバーライドオプションを設定するときは、次の考慮事項が適用されます。

- デフォルトでは、レプリケート元オブジェクトの所有者がレプリカの所有者となります。Amazon S3 は、オブジェクトバージョンとそれに関連付けられた ACL をレプリケートします。

  所有者オーバーライドを追加すると、Amazon S3 はオブジェクトのバージョンのみをレプリケートし、ACL はレプリケートしません。さらに、Amazon S3 は、以降の変更をレプリケート元オブジェクト ACL にレプリケートしません。Amazon S3 は、完全な制御をレプリケート先バケット所有者に許可する ACL をレプリカに設定します。
- 所有者オーバーライドを有効、または無効にするようにレプリケーション設定を更新すると、以下の処理が行われます。

  - レプリケーション設定へ所有者オーバーライドオプションを追加する場合。

    Amazon S3 がオブジェクトバージョンをレプリケートすると、レプリケート元オブジェクトに関連付けられている ACL は破棄されます。代わりに、ACL は完全なコントロールをレプリケート先バケットの所有者に与えるレプリカに設定されます。レプリケート元オブジェクト ACL へのそれ以降の変更はレプリケートされません。ただし、この ACL 変更は、所有者オーバーライドオプションを設定する前にレプリケートされたオブジェクトバージョンには適用されません。所有者オーバーライドが設定される前にレプリケートされたレプリケート元オブジェクトの ACL 更新は、レプリケートされ続けます (オブジェクトとそのレプリカの所有者が引き続き同じであるため)。
  - レプリケーション設定から所有者オーバーライドオプションを削除する場合。

    Amazon S3 は、レプリケート元バケットと関連 ACL に表示される新しいオブジェクトを、レプリケート先バケットにレプリケートします。所有者オーバーライドを削除する前にレプリケートされたオブジェクトの場合、Amazon S3 によるオブジェクト所有権の変更が有効なため、Amazon S3 は ACL をレプリケートしません。つまり、所有者オーバーライド設定時にレプリケートされたオブジェクトバージョンに設定された ACL は、引き続きレプリケートされません。

レプリカの所有者を変更する方法

レプリケーション設定内の######### のバケットと ######### のバケットが異なる AWS アカウント によって所有されている場合、Amazon S3 に対し、レプリカの所有権を ######### のバケットを所有している AWS アカウント に変更するよう指示できます。この例では、Amazon S3 コンソールおよび AWS CLI を使用してレプリカの所有権を変更する方法について説明します。詳細については、「レプリカ所有者の変更」を参照してください。

> **Note**
>
> S3 レプリケーションを使用し、レプリケート元バケットとレプリケート先バケットが異なった AWS アカウント によって所有される場合、バケットを使用する場合、レプリケート先バケットのバケット所有者は ACL を無効にして ([オブジェクト所有権] に対するバケット所有者の強制設定)、レプリカの所有権を送信先バケットを所有する AWS アカウント に変更することができます。この設定は、`s3:ObjectOwnerOverrideToBucketOwner` の許可の必要なく、既存の所有者のオーバーライドの動作を模倣します。つまり、バケット所有者の強制設定を使用してレプリケート先バケットにレプリケートされるすべてのオブジェクトは、レプリケート先バケット所有者によって所有されることを意味しています。オブジェクト所有権については、オブジェクトの所有権の制御とバケットの ACL の無効化。 を参照してください。

クロスアカウントシナリオで AWS Key Management Service でのサーバー側の暗号化を使用したレプリケーションの設定の詳細については、クロスアカウントシナリオに対する追加のアクセス許可の付与 を参照してください。

S3 コンソールの使用

手順については、「同じアカウントが所有するレプリケート元バケットとレプリケート先バケットのレプリケーションの設定」を参照してください。このトピックでは、バケットが同一の、または異なる AWS アカウントで所有されている場合にレプリケーション設定を行う手順について説明します。

AWS CLI の使用

AWS CLI を使用してレプリカの所有権を変更するには、バケットを作成し、バケットのバージョニングを有効にします。さらに、IAM ロールを作成して Amazon S3 にオブジェクトをレプリケートする許可を与え、レプリケート元バケットにレプリケーション設定を追加します。レプリケーション設定で、Amazon S3 にレプリカ所有者の変更を指示します。また、セットアップをテストします。

レプリケート元とレプリケート先のバケットが、異なる AWS アカウントによって所有されている場合にレプリカの所有権を変更するには (AWS CLI)

1. この例では、*`source`* および *`destination`* バケットを 2 つの異なる AWS アカウント で作成します。2 つの名前付きプロファイルで AWS CLI を設定します。この例では、それぞれ `acctA` と `acctB` という名前のプロファイルを使用します。認証情報プロファイルの設定については、AWS Command Line Interface ユーザーガイドの名前付きプロファイルを参照してください。

> **Important**
>
> この演習に使用するプロファイルは、必要なアクセス権限を持っている必要があります。たとえば、レプリケーション設定で、Amazon S3 が引き受けることができる IAM ロールを指定します。使用するプロファイルに `iam:PassRole` アクセス権限がある場合のみ実行できます。管理者ユーザーの認証情報を使用して名前付きプロファイルを作成すると、すべてのタスクを実行できるようになります。詳細については、IAM ユーザーガイドの「AWS サービスにロールを渡すアクセス権限をユーザーに付与する」を参照してください。

これらのプロファイルが必要なアクセス権限を持っていることを確認する必要があります。たとえば、レプリケーション設定には Amazon S3 が引き受けることのできる IAM ロールが含まれています。そのような設定をバケットにアタッチするために使用する名前付きプロファイルは、`iam:PassRole` アクセス権限がある場合のみ実行できます。これらの名前付きプロファイル作成時に管理者ユーザーの認証情報を指定すると、すべての権限が付与されます。詳細については、IAM ユーザーガイドの「AWS サービスにロールを渡すアクセス権限をユーザーに付与する」を参照してください。

2. *#######*バケットを作成してバージョニングを有効にします。この例では、米国東部 (バージニア北部) (us-east-1) リージョンに*#######*バケットを作成します。

```
aws s3api create-bucket \
--bucket source \
--region us-east-1 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket source \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. *#######*バケットを作成してバージョニングを有効にします。この例では、米国西部 (オレゴン) (us-west-2) リージョンに*#######*バケットを作成します。*#######*バケットに使用したものとは異なる AWS アカウント プロファイルを使用してください。

```
aws s3api create-bucket \
--bucket destination \
```

```
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctB
```

```
aws s3api put-bucket-versioning \
--bucket destination \
--versioning-configuration Status=Enabled \
--profile acctB
```

4. *#######*バケットポリシーにアクセス許可を追加して、レプリカの所有権を変更できるようにします。

   a. 以下のポリシーを *destination-bucket-policy*.json に保存します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "destination_bucket_policy_sid",
      "Principal": {
        "AWS": "source-bucket-owner-account-id"
      },
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete",
        "s3:ObjectOwnerOverrideToBucketOwner",
        "s3:ReplicateTags",
        "s3:GetObjectVersionTagging"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::destination/*"
      ]
    }
  ]
}
```

   b. このポリシーを*#########*バケットに位置します。

```
aws s3api put-bucket-policy --region $ {destination_region} --
bucket $ {destination} --policy file://destination_bucket_policy.json
```

5. IAM ロールを作成します。#######バケットに後で追加するレプリケーション設定でこのロールを指定します。Amazon S3 は、ユーザーに代わってオブジェクトをレプリケートするこのロールを引き受けます。IAM ロールは 2 つのステップで作成します。

   - ロールを作成します。
   - アクセス権限ポリシーをロールにアタッチします。

   a. IAM ロールを作成します。

      i. 次の信頼ポリシーをコピーして、ローカルコンピュータの現在のディレクトリにある s3-role-trust-policy.json という名前のファイルに保存します。このポリシーは、Amazon S3 がロールを引き受けるためのアクセス許可を付与します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

      ii. 次の AWS CLI コマンドを実行して、ロールを作成します。

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
--profile acctA
```

   b. アクセス権限ポリシーをロールにアタッチします。

      i. 次のアクセス権限ポリシーをコピーして、ローカルコンピュータの現在のディレクトリにある s3-role-perm-pol-changeowner.json という名前のファイルに保存します。このポリシーは、さまざまな Amazon S3 バケットとオブジェクトアクションに対するアクセス許可を付与します。次の手順では、IAM ロールを作成し、このポリシーをロールにアタッチします。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::source/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::source"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ObjectOwnerOverrideToBucketOwner",
            "s3:ReplicateTags",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":"arn:aws:s3:::destination/*"
      }
   ]
}
```

ii. ポリシーを作成してロールにアタッチするには、次のコマンドを実行します。

```
$ aws iam put-role-policy \
--role-name replicationRole \
--policy-document file://s3-role-perm-pol-changeowner.json \
```

```
--policy-name replicationRolechangeownerPolicy \
--profile acctA
```

6. レプリケーション設定をレプリケート元バケットに追加します。

   a. AWS CLI では、レプリケーション設定を JSON で指定する必要があります。以下の JSON を、ローカルコンピュータの現在のディレクトリにある、replication.json というファイルに保存します。設定で、レプリカ所有権の変更を示す AccessControlTranslation の追加。

```
{
   "Role":"IAM-role-ARN",
   "Rules":[
      {
         "Status":"Enabled",
         "Priority":1,
         "DeleteMarkerReplication":{
            "Status":"Disabled"
         },
         "Filter":{
         },
         "Status":"Enabled",
         "Destination":{
            "Bucket":"arn:aws:s3:::destination",
            "Account":"destination-bucket-owner-account-id",
            "AccessControlTranslation":{
               "Owner":"Destination"
            }
         }
      }
   ]
}
```

   b. *#######*バケット所有者のアカウント ID および *IAM-role-ARN* の値を指定して JSON を編集します。変更を保存します。

   c. 次のコマンドを実行して、レプリケート元バケットにレプリケーション設定を追加します。*#######*バケット名を指定してください。

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket source \
```

```
--profile acctA
```

7. Amazon S3 コンソールでレプリカの所有権を確認します。

   a. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

   b. オブジェクトを*#######*バケットに追加します。*destination* にオブジェクトレプリカが含まれていること、およびレプリカの所有権が*destination* バケットを所有する AWS アカウント に変更されていることを確認します。

AWS SDK の使用

レプリケーション設定を追加するコード例については、「AWS SDK の使用」を参照してください。レプリケーション設定を適切に変更する必要があります。概念については、「レプリカ所有者の変更」を参照してください。

S3 Replication Time Control (S3 RTC) を使用してコンプライアンス要件を満たす

S3 Replication Time Control (S3 RTC) では、データレプリケーションに関するコンプライアンス要件 (またはビジネス要件) への対応をサポートします。また、Amazon S3 レプリケーション時間を可視化します。S3 RTC は、Amazon S3 にアップロードしたほとんどのオブジェクトを数秒でレプリケートし、これらのオブジェクトの 99.99% を 15 分以内にレプリケートします。

S3 RTC にはデフォルトで S3 レプリケーションメトリクスと Amazon S3 イベント通知が含まれており、これらを使用することで、レプリケーションを保留している S3 API オペレーションの総数、レプリケーションを保留しているオブジェクトの合計サイズ、および最大レプリケーション時間をモニタリングできます。レプリケーションメトリクスは、S3 RTC とは別に有効化できます。詳細については、「レプリケーションメトリクスによる進捗状況のモニタリング」を参照してください。さらに、S3 RTC は、オブジェクトレプリケーションが 15 分のしきい値を超えた場合、または 15 分のしきい値の後にレプリケートされた場合に、バケット所有者に通知する `OperationMissedThreshold` および `OperationReplicatedAfterThreshold` イベントを提供します。

S3 RTC を使用すると、まれなケースとして、オブジェクトが 15 分以内にレプリケートされなかったとき、またこれらのオブジェクトが 15 分のしきい値の後にレプリケートされたときに、Amazon S3 イベントから通知を受け取ることができます。Amazon S3 イベントは、Amazon SQS、Amazon SNS、または AWS Lambda.を通じて利用できます。詳細については、「the section called “Amazon S3 イベント通知”」を参照してください。

トピック

- S3 Replication Time Control
- S3 RTC でのレプリケーションメトリクス
- Amazon S3 イベント通知の使用によるレプリケーションオブジェクトの追跡
- S3 RTC のベストプラクティスとガイドライン
- S3 Replication Time Control (S3 RTC) の有効化

S3 Replication Time Control

新しいレプリケーションルールまたは既存のレプリケーションルールで S3 Replication Time Control (S3 RTC) の使用を開始できます。レプリケーションルールを S3 バケット全体に適用するか、特定のプレフィックスやタグを持つ Amazon S3 オブジェクトに適用するかを選択できます。S3 RTC を有効にすると、レプリケーションルールでレプリケーションメトリクスも有効になります。

最新バージョンのレプリケーション設定を使用している (すなわち、レプリケーション設定ルールで`Filter`要素を指定している) 場合、Amazon S3 は削除マーカーをレプリケートしません。ただし、タグベース以外のルールには削除マーカーレプリケーションを追加できます。

> **Note**
>
> レプリケーションメトリクスは、Amazon CloudWatch カスタムメトリクスと同じレートで請求されます。詳細については、「Amazon CloudWatch の料金」を参照してください。

S3 RTC を使用したルールの作成の詳細については、S3 Replication Time Control (S3 RTC) の有効化を参照してください。

S3 RTC でのレプリケーションメトリクス

S3 Replication Time Control (S3 RTC) が有効なレプリケーションルールは、レプリケーションメトリクスをパブリッシュします。レプリケーションメトリクスを使用すると、レプリケーションを保留している S3 API オペレーションの合計数、レプリケーションを保留しているオブジェクトの合計サイズ、およびレプリケート先のリージョンへの最大レプリケーション時間、およびレプリケーションに失敗したオペレーションの合計数をモニタリングできます。次に、個別にレプリケートする各データセットをモニタリングできます。

レプリケーションメトリクスは、S3 RTC が有効になってから 15 分以内に使用できます。レプリケーションメトリクスは、Amazon S3 コンソール、Amazon S3 API、AWS SDK、AWS Command

Line Interface (AWS CLI)、および Amazon CloudWatch を通じてご利用いただけます。詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。

Amazon S3 コンソールを使用したレプリケーションメトリクスの検索の詳細については、Amazon S3 コンソールを使用したレプリケーションメトリクスの表示を参照してください。

Amazon S3 イベント通知の使用によるレプリケーションオブジェクトの追跡

S3 Replication Time Control (S3 RTC) が発行する特定のイベント通知をモニタリングすることで、15 分以内にレプリケートされなかったオブジェクトのレプリケーション時間を追跡できます。これらのイベントは、S3 RTC を使用したレプリケーションの対象であったオブジェクトが 15 分以内にレプリケートされなかったときや、そのオブジェクトが 15 分のしきい値の後にレプリケートされたときに発行されます。

レプリケーションイベントは、S3 RTC が有効になってから 15 分以内に使用できます。Amazon S3 イベントは、Amazon SQS、Amazon SNS、または AWS Lambda.を通じて利用できます。詳細については、「Amazon S3 イベント通知」を参照してください。

S3 RTC のベストプラクティスとガイドライン

S3 Replication Time Control (S3 RTC) を使用して Amazon S3 にデータをレプリケートする場合は、以下のベストプラクティスガイドラインに従って、ワークロードのレプリケーションパフォーマンスを最適化します。

トピック

- Amazon S3 レプリケーションおよびリクエストレートのパフォーマンスガイドライン
- レプリケーションリクエストレートの見積り
- S3 RTC データ転送レート制限を超過する場合
- AWS KMS 暗号化オブジェクトのレプリケーションリクエストレート

Amazon S3 レプリケーションおよびリクエストレートのパフォーマンスガイドライン

アプリケーションは、Amazon S3 レプリケーション のストレージをアップロードおよび取得する際に、リクエストパフォーマンスで 1 秒あたり何千ものトランザクションを達成できます。たとえば、アプリケーションは S3 バケットでプレフィックスあたり毎秒 3,500 個以上の PUT/COPY/POST/DELETE リクエストまたは 5,500 個以上の GET/HEAD リクエストを処理できます。これには、S3 レプリケーションがユーザーに代わって行うリクエストも含まれます。バケット内のプレフィックスの数に制限はありません。読み取りを並列化することによって読み取りまたは書き込みの

パフォーマンスを向上させることができます。たとえば、S3 バケットに 10 個のプレフィックスを作成して読み取りを並列化すると、読み取りパフォーマンスを 1 秒あたり 55,000 回の読み取りリクエストにスケールできます。

Amazon S3 は、これらのガイドラインを上回る持続的なリクエストレートや、LIST リクエストで同時実行される持続的なリクエストレートに応じて自動的にスケールします。Amazon S3 が新しいリクエストレートに合わせて最適化を行っている間、最適化が完了するまで一時的に HTTP 503 リクエストレスポンスが送信される場合があります。これは、1 秒あたりのリクエスト数が増加したときや、S3 RTC を初めて有効にしたときに発生する場合があります。この間は、レプリケーションのレイテンシーが増加する可能性があります。S3 RTC サービスレベルアグリーメント (SLA) は、1 秒あたりのリクエストに関する Amazon S3 パフォーマンスガイドラインを超えた期間には適用されません。

S3 RTC SLA は、レプリケーションデータ転送速度がデフォルトの 1 Gbps 制限を超えた期間にも適用されません。レプリケーション転送速度が 1 Gbps を超えることが予想される場合は、AWS Support センターに問い合わせるか、Service Quotas を使用して制限の引き上げをリクエストできます。

レプリケーションリクエストレートの見積り

合計リクエストレート (Amazon S3 レプリケーションがユーザーに代わって行うリクエストを含む) は、レプリケート元バケットとレプリケート先バケットの両方で Amazon S3 リクエストレートのガイドライン内に収まる必要があります。レプリケートされたオブジェクトごとに、Amazon S3 レプリケーションがレプリケート元に最大 5 件の GET/HEAD リクエストと 1 件の PUT リクエストを行い、レプリケート先バケットに 1 件の PUT リクエストを行います。

たとえば、1 秒あたり 100 個のオブジェクトをレプリケートする場合、Amazon S3 レプリケーションはユーザーに代わって 100 件の PUT リクエストを追加で実行する可能性があります。つまり、レプリケート元 S3 バケットに対して 1 秒あたり合計 200 個の PUT リクエストが実行されることになります。Amazon S3 レプリケーションでは、最大 500 件の GET/HEAD (レプリケートされたオブジェクトごとに 5 件の GET/HEAD リクエスト) を実行する可能性があります。

> **Note**
>
> レプリケートされたオブジェクトごとに 1 つの PUT リクエストに対してのみコストが発生します。詳細については、「Amazon S3 のレプリケーションに関するよくある質問」で料金情報を参照してください。

S3 RTC データ転送レート制限を超過する場合

S3 レプリケーション時間コントロールのデータ転送速度がデフォルトの 1 Gbps の制限を超えると予想される場合は、AWS Support センターに問い合わせるか、Service Quotas を使用して、制限の引き上げをリクエストしてください。

AWS KMS 暗号化オブジェクトのレプリケーションリクエストレート

Amazon S3 レプリケーションを使用してサーバー側の暗号化 (SSE-KMS) で暗号化されたオブジェクトをレプリケートする場合、AWS Key Management Service (AWS KMS) リクエスト/秒の制限が適用されます。AWS KMS は、リクエストレートが 1 秒あたりのリクエスト数の制限を超えているために (そうでなければ) 有効なリクエストを拒否することがあります。リクエストがスロットリングされると、AWS KMS は `ThrottlingException` エラーを返します。AWS KMS リクエストレートの制限は、ユーザーが直接行うリクエストと、ユーザーに代わって Amazon S3 レプリケーションが行うリクエストに適用されます。

例えば、1 秒あたり 1,000 個のオブジェクトをレプリケートする場合、AWS KMS リクエストレートの制限から 2,000 個のリクエストを差し引くことができます。差し引いた後の 1 秒あたりのリクエストレートを、レプリケーションを除く AWS KMS ワークロードで使用できます。AWS KMSAmazon CloudWatch でリクエストメトリクスを使用して、AWS アカウント の合計 AWS KMS リクエストレートをモニタリングできます。

S3 Replication Time Control (S3 RTC) の有効化

S3 Replication Time Control (S3 RTC) では、データレプリケーションに関するコンプライアンス要件 (またはビジネス要件) への対応をサポートします。また、Amazon S3 レプリケーション時間を可視化します。S3 RTC は、Amazon S3 にアップロードしたほとんどのオブジェクトを数秒でレプリケートし、これらのオブジェクトの 99.99% を 15 分以内にレプリケートします。

S3 RTC を使用すると、レプリケーションが保留中のオブジェクトの合計数とサイズ、およびレプリケート先リージョンへの最大レプリケーション時間をモニタリングできます。レプリケーションのメトリクスは、AWS Management ConsoleおよびAmazon CloudWatch ユーザーガイドから入手できます。詳細については、「the section called “CloudWatch の S3 レプリケーションメトリクス”」を参照してください。

S3 コンソールの使用

手順については、「同じアカウントが所有するレプリケート元バケットとレプリケート先バケットのレプリケーションの設定」を参照してください。このトピックでは、バケットが同じ AWS アカウン

トや異なるアカウントで所有されている場合に、レプリケーション設定で S3 RTC を有効にする手順について説明します。

AWS CLI の使用

AWS CLI を使用して S3 RTC が有効になったオブジェクトをレプリケートするには、バケットを作成し、バケットのバージョニングを有効にします。さらに、IAM ロールを作成して Amazon S3 にオブジェクトをレプリケートする許可を与え、レプリケート元バケットにレプリケーション設定を追加します。レプリケーション設定で S3 Replication Time Control (S3 RTC) を有効にする必要があります。

S3 RTC を有効にしてレプリケートするには (AWS CLI)

- 次の例ではReplicationTimeおよびMetricを設定し、レプリケート元バケットにレプリケーション設定を追加します。

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
                "Status": "Disabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::destination",
                "Metrics": {
                    "Status": "Enabled",
                    "EventThreshold": {
                        "Minutes": 15
                    }
                },
                "ReplicationTime": {
                    "Status": "Enabled",
                    "Time": {
                        "Minutes": 15
                    }
                }
            },
            "Priority": 1
        }
```

```
    ],
    "Role": "IAM-Role-ARN"
}
```

> **Important**
>
> `Metrics:EventThreshold:Minutes`と`ReplicationTime:Time:Minutes`に設定できる唯一の有効な値は 15 です。

AWS SDK for Java の使用

S3 Replication Time Control (S3 RTC) でレプリケーション設定を追加する Java の例は、次のとおりです。

```
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.DeleteMarkerReplication;
import software.amazon.awssdk.services.s3.model.Destination;
import software.amazon.awssdk.services.s3.model.Metrics;
import software.amazon.awssdk.services.s3.model.MetricsStatus;
import software.amazon.awssdk.services.s3.model.PutBucketReplicationRequest;
import software.amazon.awssdk.services.s3.model.ReplicationConfiguration;
import software.amazon.awssdk.services.s3.model.ReplicationRule;
import software.amazon.awssdk.services.s3.model.ReplicationRuleFilter;
import software.amazon.awssdk.services.s3.model.ReplicationTime;
import software.amazon.awssdk.services.s3.model.ReplicationTimeStatus;
import software.amazon.awssdk.services.s3.model.ReplicationTimeValue;

public class Main {

  public static void main(String[] args) {
    S3Client s3 = S3Client.builder()
      .region(Region.US_EAST_1)
      .credentialsProvider(() -> AwsBasicCredentials.create(
          "AWS_ACCESS_KEY_ID",
          "AWS_SECRET_ACCESS_KEY")
      )
      .build();

    ReplicationConfiguration replicationConfig = ReplicationConfiguration
      .builder()
```

```
      .rules(
          ReplicationRule
            .builder()
            .status("Enabled")
            .priority(1)
            .deleteMarkerReplication(
                DeleteMarkerReplication
                    .builder()
                    .status("Disabled")
                    .build()
            )
            .destination(
                Destination
                    .builder()
                    .bucket("destination_bucket_arn")
                    .replicationTime(
                        ReplicationTime.builder().time(
                            ReplicationTimeValue.builder().minutes(15).build()
                        ).status(
                            ReplicationTimeStatus.ENABLED
                        ).build()
                    )
                    .metrics(
                        Metrics.builder().eventThreshold(
                            ReplicationTimeValue.builder().minutes(15).build()
                        ).status(
                            MetricsStatus.ENABLED
                        ).build()
                    )
                    .build()
            )
            .filter(
                ReplicationRuleFilter
                    .builder()
                    .prefix("testtest")
                    .build()
            )
        .build())
        .role("role_arn")
        .build();

    // Put replication configuration
    PutBucketReplicationRequest putBucketReplicationRequest =
PutBucketReplicationRequest
```

```
        .builder()
        .bucket("source_bucket")
        .replicationConfiguration(replicationConfig)
        .build();

    s3.putBucketReplication(putBucketReplicationRequest);
  }
}
```

詳細については、「S3 Replication Time Control (S3 RTC) を使用してコンプライアンス要件を満たす」を参照してください。

暗号化されたオブジェクトのレプリケート (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

サーバー側の暗号化を使用して暗号化されているオブジェクトをレプリケートする場合は、いくつかの特別な考慮事項があります。Amazon S3 は、以下の 3 種類のサーバー側の暗号化をサポートしています。

- Amazon S3 マネージドキーを用いたサーバー側の暗号化 (SSE-S3)
- AWS Key Management Service (AWS KMS) キー (SSE-KMS) によるサーバー側の暗号化
- AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS)
- 顧客提供のキーを用いたサーバー側の暗号化 (SSE-C)。

サーバーサイドの暗号化の詳細については、「the section called “サーバー側の暗号化”」を参照してください。

このトピックでは、サーバー側の暗号化を使用して暗号化されているオブジェクトをレプリケートするよう Amazon S3 に指示するために必要なアクセス許可について説明します。このトピックでは、追加できるその他の設定要素や、暗号化されたオブジェクトのレプリケーションに必要なアクセス許可を付与する AWS Identity and Access Management (IAM) ポリシーの例も提供します。

ステップバイステップの手順と例については、「暗号化されたオブジェクトのレプリケーションの有効化」を参照してください。レプリケーション設定の作成については、「オブジェクトのレプリケーション」を参照してください。

> **Note**
>
> Amazon S3 では、マルチリージョン AWS KMS keys を使用できます。ただし、Amazon S3 では現在、マルチリージョンキーは、単一リージョンキーであるかのように処理され、キーのマルチリージョン特徴は使用しません。詳細については、「AWS Key Management Service デベロッパーガイド」の「マルチリージョンキーを使用する」を参照してください。

トピック

バケットのデフォルトの暗号化がレプリケーションに与える影響

レプリケーション先バケットのデフォルトの暗号化を有効にすると、以下の暗号化動作が適用されます。

- レプリケート元バケットのオブジェクトが暗号化されていない場合、レプリケート先バケットのレプリカオブジェクトはレプリケート先バケットのデフォルトの暗号化設定を使用して暗号化されます。そのため、レプリケート元のオブジェクトのエンティティタグ (ETag) はレプリカオブジェクトの ETag とは異なります。アプリケーションで ETag を使用している場合は、アプリケーションを更新して、この違いを反映する必要があります。

- レプリケート元バケット内のオブジェクトが Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)、AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS)、または AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS) を使用して暗号化されている場合、レプリケート先バケットのレプリカオブジェクトは、レプリケート元オブジェクトと同じタイプの暗号化を使用します。レプリケート先バケットのデフォルトの暗号化設定は使用されません。

SSE-C で暗号化されたオブジェクトのレプリケーション

お客様が指定したキーによるサーバー側の暗号化 (SSE-C) を使用することで、独自の暗号化キーを管理できます。SSE-C ではユーザーがキーを管理し、Amazon S3 は暗号化と復号のプロセスを管理します。ユーザーはリクエストの一部として暗号化キーを指定する必要がありますが、オブジェクトの暗号化または復号を実行するコードを記述する必要はありません。オブジェクトをアップロードすると、Amazon S3 は指定されたキーを使用してオブジェクトを暗号化します。その後、Amazon S3 はそのキーをメモリから削除します。オブジェクトを取得するときは、リクエストの中で同じ暗号化キーを指定する必要があります。詳細については、「the section called “お客様が指定したキーによるサーバー側の暗号化 (SSE-C)”」を参照してください。

S3 レプリケーションは SSE-C で暗号化されたオブジェクトをサポートしています。SSE-C オブジェクトのレプリケーションは、暗号化されていないオブジェクトのレプリケーションを設定するのと同じ方法で、Amazon S3 コンソールまたは AWS SDK で設定できます。レプリケーションに現在必要な権限以外に SSE-C 権限を追加することはできません。

S3 レプリケーションは、新しくアップロードされた SSE-C 暗号化オブジェクトを、S3 レプリケーション設定に従って自動的にレプリケートします (該当する場合)。バケット内の既存のオブジェクトをレプリケートするには、S3 バッチレプリケーションを使用します。オブジェクトのレプリケーションの詳細については、「the section called “ライブレプリケーションの設定”」および「the section called “既存のオブジェクトのレプリケーション”」を参照してください。

SSE-C オブジェクトのレプリケートに追加料金はかかりません。レプリケーション料金の詳細については、「Amazon S3 の料金ページ」を参照してください。

SSE-S3、SSE-KMS、または DSSE-KMS で暗号化されたオブジェクトのレプリケーション

デフォルトでは、Amazon S3 は SSE-KMS または DSSE-KMS で暗号化されたオブジェクトをレプリケートしません。このセクションでは、これらのオブジェクトをレプリケートするよう Amazon S3 に指示するために追加できるその他の設定要素について説明します。

ステップバイステップの手順と例については、「暗号化されたオブジェクトのレプリケーションの有効化」を参照してください。レプリケーション設定の作成については、「オブジェクトのレプリケーション」を参照してください。

レプリケーション設定の追加情報の指定

レプリケーション設定で、以下を実行してください。

- 次のレプリケーション設定例に示すように、レプリケーション設定の Destination 要素に、オブジェクトレプリカを暗号化するために Amazon S3 で使用する対称 AWS KMS カスタマーマネージドキーの ID を追加します。
- KMS キー (SSE-KMS または DSSE-KMS) を使用して暗号化されたオブジェクトのレプリケーションを有効にすることで、明示的にオプトインします。オプトインするには、次のレプリケーション設定例に示すように、SourceSelectionCriteria 要素を追加します。

```
<ReplicationConfiguration>
   <Rule>
      ...
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
           <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>

      <Destination>
          ...
          <EncryptionConfiguration>
             <ReplicaKmsKeyID>AWS KMS key ARN or Key Alias ARN that's in the same AWS
 ##### as the destination bucket.</ReplicaKmsKeyID>
          </EncryptionConfiguration>
       </Destination>
      ...
   </Rule>
</ReplicationConfiguration>
```

> **Important**
>
> KMS キーは、レプリケート先バケットと同じ AWS リージョン 内で作成されている必要があります。

KMS キーは有効 である必要があります。PutBucketReplication API オペレーションは、KMS キーの有効性を確認しません。無効な KMS キーを使用した場合、応答として HTTP 200 OK ステータスコードを受け取りますが、レプリケーションは失敗します。

次の例は、オプションの設定要素が含まれているレプリケーション設定を示します。このレプリケーション設定にはルールが 1 つあります。このルールは、Tax キープレフィックスが付いているオブジェクトに適用されます。Amazon S3 は、指定された AWS KMS key ID を使用して、これらのオブジェクトレプリカを暗号化します。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration>
   <Role>arn:aws:iam::account-id:role/role-name</Role>
   <Rule>
      <ID>Rule-1</ID>
      <Priority>1</Priority>
      <Status>Enabled</Status>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::example-s3-destination-bucket</Bucket>
         <EncryptionConfiguration>
            <ReplicaKmsKeyID>AWS KMS key ARN or Key Alias ARN that's in the same AWS
 ##### as the destination bucket. (S3 uses this key to encrypt object replicas.)</
ReplicaKmsKeyID>
         </EncryptionConfiguration>
      </Destination>
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
            <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>
   </Rule>
</ReplicationConfiguration>
```

IAM ロールに対する追加のアクセス許可の付与

SSE-S3、SSE-KMS、または DSSE-KMS を使用して保管時に暗号化されたオブジェクトをレプリケートするには、レプリケーション設定で指定した AWS Identity and Access Management (IAM) ロールに以下の追加のアクセス許可を付与します。IAM ロールに関連付けられているアクセス許可ポリシーを更新してこれらの権限を付与します。

- レプリケート元オブジェクトのための **`s3:GetObjectVersionForReplication`** アクション – このアクションにより、Amazon S3 は、暗号化されていないオブジェクトと、SSE-S3、SSE-KMS、または DSSE-KMS を使用したサーバー側の暗号化で作成されたオブジェクトの両方をレプリケートできるようになります。

  > **Note**
  >
  > `s3:GetObjectVersion` アクションの代わりに `s3:GetObjectVersionForReplication` アクションの使用をお勧めします。`s3:GetObjectVersionForReplication` ではレプリケーションに必要な最小限のアクセス許可のみを Amazon S3 に提供するためです。さらに、`s3:GetObjectVersion` アクションにより、暗号化されていないオブジェクトや SSE-S3 で暗号化されたオブジェクトはレプリケートできますが、KMS キー (SSE-KMS または DSSE-KMS) を使用して暗号化されたオブジェクトをレプリケートすることはできません。

- KMS キーの **`kms:Decrypt`** および **`kms:Encrypt`** AWS KMS アクション
  - レプリケート元オブジェクトの復号に使用された AWS KMS keyに対して `kms:Decrypt` 権限を付与する必要があります。
  - オブジェクトレプリカの暗号化に使用された AWS KMS keyに対して `kms:Encrypt` 許可を付与する必要があります。
- プレーンテキストオブジェクトをレプリケートするための **`kms:GenerateDataKey`** アクション — SSE-KMS または DSSE-KMS 暗号化がデフォルトで有効になっているバケットにプレーンテキストオブジェクトをレプリケートする場合は、レプリケート先の暗号化コンテキストと KMS キーの `kms:GenerateDataKey` アクセス許可を IAM ポリシーに含める必要があります。

AWS KMS 条件キーを使用して、これらの許可を、レプリケート先バケットとオブジェクトのみに制限することをお勧めします。IAM ロールを所有する AWS アカウントには、ポリシーにリストされている KMS キーに対する `kms:Encrypt` アクションおよび `kms:Decrypt` アクションのアクセス許可が必要です。KMS キーが別の AWS アカウント によって所有されている場合は、KMS キーの所有者が IAM ロールを所有する AWS アカウントにこれらの許可を付与する必要があります。これらの

KMS キーへのアクセス管理の詳細については、 AWS Key Management Service デベロッパーガイドのAWS KMS での IAM ポリシーの使用を参照してください。

S3 バケットキーとレプリケーション

S3 バケットキーでレプリケーションを使用するには、オブジェクトレプリカの暗号化に使用する KMS キーの AWS KMS keyポリシーに、呼び出し元のプリンシパルの `kms:Decrypt` 許可を含める必要があります。`kms:Decrypt` の呼び出しでは、S3 バケットキーを使用する前にその整合性を検証します。詳細については、「レプリケーションでの S3 バケットキーの使用」を参照してください。

レプリケート元バケットまたはレプリケート先バケットで S3 バケットキーを有効にすると、暗号化コンテキストはバケットの Amazon リソースネーム (ARN) になり、オブジェクトの ARN (`arn:aws:s3:::`*`bucket_ARN`* など) にはなりません。IAM ポリシーを更新して、暗号化コンテキストにバケット ARN を使用する必要があります。

```
"kms:EncryptionContext:aws:s3:arn": [
"arn:aws:s3:::bucket_ARN"
]
```

詳細については、「暗号化コンテキスト (x-amz-server-side-encryption-context)」(「REST API の使用」セクション) と「S3 バケットキーを有効にする前に注意するべき変更点」を参照してください。

ポリシーの例 – レプリケーションに SSE-S3 と SSE-KMS を使用する

次の IAM ポリシーの例は、SSE-S3 と SSE-KMS をレプリケーションで使用するためのステートメントを示しています。

Example – 個別のレプリケート先バケットで SSE-KMS を使用する

次のポリシーの例は、個別のレプリケート先バケットで SSE-KMS を使用するためのステートメントを示しています。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action": ["kms:Decrypt"],
         "Effect": "Allow",
         "Condition": {
             "StringLike": {
```

```
                "kms:ViaService": "s3.source-bucket-region.amazonaws.com",
                "kms:EncryptionContext:aws:s3:arn": [
                    "arn:aws:s3:::example-s3-source-bucket/key-prefix1*"
                ]
            }
         },
         "Resource": [
            "List of AWS KMS key ARNs that are used to encrypt source objects."
         ]
      },
      {
         "Action": ["kms:Encrypt"],
         "Effect": "Allow",
         "Condition": {
            "StringLike": {
               "kms:ViaService": "s3.destination-bucket-1-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn": [
                  "arn:aws:s3:::example-s3-destination-bucket1/key-prefix1*"
               ]
         }
         },
         "Resource": [
            "AWS KMS key ARNs (in the same AWS ##### as destination bucket 1). Used to
encrypt object replicas created in destination bucket 1."
         ]
      },
      {
         "Action": ["kms:Encrypt"],
         "Effect": "Allow",
         "Condition": {
            "StringLike": {
               "kms:ViaService": "s3.destination-bucket-2-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn": [
                  "arn:aws:s3:::example-s3-destination-bucket2/key-prefix1*"
               ]
         }
         },
         "Resource": [
            "AWS KMS key ARNs (in the same AWS ##### as destination bucket 2). Used to
encrypt object replicas created in destination bucket 2."
         ]
      }
   ]
```

```
}
```

Example - SSE-S3 と SSE-KMS で作成したオブジェクトのレプリケーション

以下は、暗号化されていないオブジェクト、SSE-S3 で作成したオブジェクト、および SSE-KMS で作成したオブジェクトをレプリケートするのに必要な許可を付与する完全な IAM ポリシーを示しています。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::example-s3-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::example-s3-source-bucket/key-prefix1*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete"
         ],
         "Resource":"arn:aws:s3:::example-s3-destination-bucket/key-prefix1*"
      },
      {
         "Action":[
            "kms:Decrypt"
         ],
```

```
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.source-bucket-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::example-s3-source-bucket/key-prefix1*"
               ]
            }
         },
         "Resource":[
           "List of the AWS KMS key ARNs that are used to encrypt source objects."
         ]
      },
      {
         "Action":[
            "kms:Encrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.destination-bucket-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::example-s3-destination-bucket/prefix1*"
               ]
            }
         },
         "Resource":[
            "AWS KMS key ARNs (in the same AWS ##### as the destination bucket) to use
 for encrypting object replicas"
         ]
      }
   ]
}
```

Example – S3 バケットキーを使用したオブジェクトのレプリケーション

次に示すのは、S3 バケットキーを使用してオブジェクトをレプリケートするのに必要なアクセス許可を付与する完全な IAM ポリシーです。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
```

```
            "Effect":"Allow",
            "Action":[
               "s3:GetReplicationConfiguration",
               "s3:ListBucket"
            ],
            "Resource":[
               "arn:aws:s3:::example-s3-source-bucket"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:GetObjectVersionForReplication",
               "s3:GetObjectVersionAcl"
            ],
            "Resource":[
               "arn:aws:s3:::example-s3-source-bucket/key-prefix1*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:ReplicateObject",
               "s3:ReplicateDelete"
            ],
            "Resource":"arn:aws:s3:::example-s3-destination-bucket/key-prefix1*"
         },
         {
            "Action":[
               "kms:Decrypt"
            ],
            "Effect":"Allow",
            "Condition":{
               "StringLike":{
                  "kms:ViaService":"s3.source-bucket-region.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn":[
                     "arn:aws:s3:::example-s3-source-bucket"
                  ]
               }
            },
            "Resource":[
              "List of the AWS KMS key ARNs that are used to encrypt source objects."
            ]
         },
```

```
      {
         "Action":[
            "kms:Encrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.destination-bucket-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::example-s3-destination-bucket"
               ]
            }
         },
         "Resource":[
            "AWS KMS key ARNs (in the same AWS ##### as the destination bucket) to use
 for encrypting object replicas"
         ]
      }
   ]
}
```

クロスアカウントシナリオに対する追加のアクセス許可の付与

レプリケート元バケットとレプリケート先バケットが、異なる AWS アカウントによって所有されているクロスアカウントシナリオでは、KMS キーを使用してオブジェクトレプリカを暗号化できます。ただし、KMS キー所有者は、レプリケート元バケットの所有者に KMS キーを使用する許可を付与する必要があります。

> **Note**
>
> SSE-KMS データをクロスアカウントでレプリケートする必要がある場合、レプリケーションルールでは、レプリケート先アカウントの AWS KMS からのカスタマーマネージドキーを指定する必要があります。AWS マネージドキー はクロスアカウントの使用を許可しないため、クロスアカウントレプリケーションの実行には使用できません。

レプリケート元バケット所有者に KMS キー (AWS KMS コンソール) を使用する許可を付与するには

1. AWS Management Console にサインインし、AWS KMS コンソール (https://console.aws.amazon.com/kms) を開きます。
2. AWS リージョン を変更するには、ページの右上隅にあるリージョンセレクターを使用します。

3. ユーザーが作成および管理するアカウント内のキーを表示するには、ナビゲーションペインで [Customer managed keys] (カスタマーマネージドキー) を選択します。
4. KMS キーを選択します。
5. [一般設定] セクションで、[キーポリシー] タブを選択します。
6. [別の AWS アカウント] まで下にスクロールします。
7. [別の AWS アカウントを追加] を選択します。

   [別の AWS アカウント] ダイアログボックスが表示されます。
8. ダイアログボックスで、[別の AWS アカウントを追加] を選択します。[arn:aws:iam::] に、レプリケート元バケットのアカウント ID を入力します。
9. [変更の保存] をクリックします。

レプリケート元バケット所有者に KMS キー(AWS CLI) を使用する許可を付与するには

- `put-key-policy` AWS Command Line Interface (AWS CLI) コマンドの詳細については、「AWS CLI コマンドリファレンス」の「put-key-policy」を参照してください。基盤となる `PutKeyPolicy` API オペレーションの詳細については、「AWS Key Management Service API リファレンス」の「PutKeyPolicy」を参照してください。

AWS KMS トランザクションクォータに関する考慮事項

クロスリージョンレプリケーション (CRR) を有効にした後で、AWS KMS 暗号化を使用して多数の新しいオブジェクトを追加すると、スロットリング (HTTP `503 Service Unavailable` エラー) が発生する可能性があります。1 秒あたりの AWS KMS トランザクション数が現在のクォータを超えると、スロットリングが発生します。詳細については、「AWS Key Management Service デベロッパーガイド」の「クォータ」を参照してください。

クォータの引き上げをリクエストするには、Service Quotas を使用してください。詳細については、「Requesting a quota increase」(クォータ引き上げのリクエスト) を参照してください。リージョンの Service Quotas がサポートされていない場合は、AWS Support ケースを開きます。

暗号化されたオブジェクトのレプリケーションの有効化

デフォルトでは、Amazon S3 は AWS Key Management Service (AWS KMS) キーによるサーバー側の暗号化 (SSE-KMS) または AWS KMS キーによる二層式サーバー側の暗号化 (DSSE-KMS) を使用して暗号化されたオブジェクトをレプリケートしません。SSE-KMS または DSSE-KMS で暗号化されたオブジェクトをレプリケートするには、バケットレプリケーション設定を変更して、Amazon

S3 がこれらのオブジェクトをレプリケートするように指示します。この例では、Amazon S3 コンソールと AWS Command Line Interface (AWS CLI) を使用してバケットのレプリケーション設定を変更し、暗号化オブジェクトのレプリケーションを有効にする方法について説明します。

詳細については、「暗号化されたオブジェクトのレプリケート (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)」を参照してください。

> **Note**
>
> レプリケート元バケットまたはレプリケート先バケットで S3 バケットキーを有効にすると、暗号化コンテキストはバケットの Amazon リソースネーム (ARN) になり、オブジェクトの ARN にはなりません。暗号化コンテキストとしてバケット ARN を使用するには、IAM ポリシーを更新する必要があります。詳細については、「S3 バケットキーとレプリケーション」を参照してください。

> **Note**
>
> Amazon S3 では、マルチリージョン AWS KMS keys を使用できます。ただし、Amazon S3 では現在、マルチリージョンキーは、単一リージョンキーであるかのように処理され、キーのマルチリージョン特徴は使用しません。詳細については、「AWS Key Management Service デベロッパーガイド」の「マルチリージョンキーを使用する」を参照してください。

S3 コンソールの使用

手順については、「同じアカウントが所有するレプリケート元バケットとレプリケート先バケットのレプリケーションの設定」を参照してください。このトピックでは、バケットが同一の、または異なる AWS アカウント によって所有されている場合にレプリケーション設定を行う手順について説明します。

AWS CLI の使用

AWS CLI で暗号化されたオブジェクトをレプリケートするには、以下の操作を実行します。

- レプリケート元バケットとレプリケート先バケットを作成して、これらのバケットのバージョニングを有効にします。

- Amazon S3 にオブジェクトをレプリケートするアクセス許可を付与する AWS Identity and Access Management (IAM) サービスロールを作成します。IAM ロールのアクセス許可には、暗号化されたオブジェクトをレプリケートするために必要なアクセス許可が含まれています。
- レプリケート元バケットにレプリケーション設定を追加します。レプリケーション設定は、KMS キーを使用して暗号化されたオブジェクトのレプリケーションに関する情報を提供します。
- レプリケート元バケットに暗号化されたオブジェクトを追加します。
- セットアップをテストして、暗号化されたオブジェクトがレプリケート先バケットにレプリケートされていることを確認します。

このプロセスを以下の手順で説明します。

サーバー側の暗号化されたオブジェクトをレプリケートするには (AWS CLI)

1. この例では、*example-s3-source-bucket* バケットと *example-s3-destination-bucket* バケットの両方を同じ AWS アカウント に作成します。AWS CLI の認証情報プロファイルも設定します。この例では、プロファイル名 *acctA* を使用します。

   認証情報プロファイルの設定については、AWS Command Line Interfaceユーザーガイドの「名前付きプロファイル」を参照してください。この例のコマンドを使用するには、*user input placeholders* を自分の情報に置き換えてください。

2. 次のコマンドを使用して *DOC-EXAMPLE-SOURCE-BUCKET* バケットを作成し、バージョニングを有効にします。次の例のコマンドは、*DOC-EXAMPLE-SOURCE-BUCKET* バケットを米国東部 (バージニア北部) (us-east-1) リージョンに作成します。

   ```
   aws s3api create-bucket \
   --bucket DOC-EXAMPLE-SOURCE-BUCKET \
   --region us-east-1 \
   --profile acctA
   ```

   ```
   aws s3api put-bucket-versioning \
   --bucket DOC-EXAMPLE-SOURCE-BUCKET \
   --versioning-configuration Status=Enabled \
   --profile acctA
   ```

3. 次のコマンドを使用して *DOC-EXAMPLE-DESTINATION-BUCKET* バケットを作成し、バージョニングを有効にします。次のコマンドは、*DOC-EXAMPLE-DESTINATION-BUCKET* バケットを米国西部 (オレゴン) (us-west-2) リージョンに作成します。

> **Note**
>
> *DOC-EXAMPLE-SOURCE-BUCKET* と *DOC-EXAMPLE-DESTINATION-BUCKET* の両方のバケットが同じ AWS アカウント にあるときに、レプリケーション設定をセットアップするには、同じプロファイルを使用します。この例では、*acctA* を使用します。バケットが異なる AWS アカウント によって所有されているときに、レプリケーションを設定するには、それぞれに異なるプロファイルを指定します。

```
aws s3api create-bucket \
--bucket DOC-EXAMPLE-DESTINATION-BUCKET \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket DOC-EXAMPLE-DESTINATION-BUCKET \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. 次に、IAM サービスロールを作成します。このロールは、後で *DOC-EXAMPLE-SOURCE-BUCKET* バケットに追加するレプリケーション設定で指定します。Amazon S3 は、ユーザーに代わってオブジェクトをレプリケートするこの ロールを引き受けます。IAM ロールは 2 つのステップで作成します。

   - サービスロールを作成します。
   - アクセス権限ポリシーをロールにアタッチします。

   a. IAM サービスロールを作成するには、以下を実行します。

      i. 次の信頼ポリシーをコピーして、ローカルコンピュータの現在のディレクトリにある `s3-role-trust-policy-kmsobj.json` という名前のファイルに保存します。このポリシーは、ロールを引き受けるサービスプリンシパルアクセス許可を Amazon S3 に付与し、Amazon S3 がユーザーに代わってタスクを実行できるようにします。

```
{
   "Version":"2012-10-17",
```

```
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

ii. 次のコマンドを使用して、ロールを作成します。

```
$ aws iam create-role \
--role-name replicationRolekmsobj \
--assume-role-policy-document file://s3-role-trust-policy-kmsobj.json  \
--profile acctA
```

b. 次に、アクセス許可ポリシーをロールにアタッチします。このポリシーは、さまざまな Amazon S3 バケットとオブジェクトアクションに対するアクセス許可を付与します。

i. 次のアクセス権限ポリシーをコピーして、ローカルコンピュータの現在のディレクトリにある s3-role-permissions-policykmsobj.json という名前のファイルに保存します。IAM ロールを作成して、後でポリシーをアタッチします。

> **Important**
>
> アクセス許可ポリシーで、*example-s3-source-bucket* および *example-s3-destination-bucket* バケットの暗号化に使用される AWS KMS キー ID を指定します。*example-s3-source-bucket* および *example-s3-destination-bucket* バケット用に 2 つの個別の KMS キーを作成する必要があります。AWS KMS keys は、それらが作成された AWS リージョン の外部では共有されません。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:ListBucket",
```

```
            "s3:GetReplicationConfiguration",
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::example-s3-source-bucket",
            "arn:aws:s3:::example-s3-source-bucket/*"
         ]
      },
      {
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLikeIfExists":{
               "s3:x-amz-server-side-encryption":[
                  "aws:kms",
                  "AES256",
                  "aws:kms:dsse"
               ],
               "s3:x-amz-server-side-encryption-aws-kms-key-id":[
                  "AWS KMS key IDs(in ARN format) to use for encrypting
object replicas"
               ]
            }
         },
         "Resource":"arn:aws:s3:::example-s3-destination-bucket/*"
      },
      {
         "Action":[
            "kms:Decrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.us-east-1.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::example-s3-source-bucket/*"
               ]
```

```
            }
         },
         "Resource":[
            "AWS KMS key IDs(in ARN format) used to encrypt source
 objects."
         ]
      },
      {
         "Action":[
            "kms:Encrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.us-west-2.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::example-s3-destination-bucket/*"
               ]
            }
         },
         "Resource":[
            "AWS KMS key IDs(in ARN format) to use for encrypting object
 replicas"
         ]
      }
   ]
}
```

ii. ポリシーを作成し、ロールにアタッチします。

```
$ aws iam put-role-policy \
--role-name replicationRolekmsobj \
--policy-document file://s3-role-permissions-policykmsobj.json \
--policy-name replicationRolechangeownerPolicy \
--profile acctA
```

5. 次に、以下のレプリケーション設定を *example-s3-source-bucket* バケットに追加します。これは、Tax/ プレフィックスを持つオブジェクトを *example-s3-destination-bucket* バケットにレプリケートするように Amazon S3 に指示します。

> **Important**
>
> レプリケーション設定で、Amazon S3 が引き受けることができる IAM ロールを指定します。ユーザーに `iam:PassRole` アクセス権限がある場合のみ実行できます。CLI コマンドで指定するプロファイルには、このアクセス許可が必要です。詳細については、IAM ユーザーガイドの「AWS のサービス のサービスにロールを渡すアクセス権限をユーザーに付与する」を参照してください。

```
 <ReplicationConfiguration>
  <Role>IAM-Role-ARN</Role>
  <Rule>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>Disabled</Status>
    </DeleteMarkerReplication>
    <Filter>
       <Prefix>Tax</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <SourceSelectionCriteria>
      <SseKmsEncryptedObjects>
        <Status>Enabled</Status>
      </SseKmsEncryptedObjects>
    </SourceSelectionCriteria>
    <Destination>
      <Bucket>arn:aws:s3:::example-s3-destination-bucket</Bucket>
      <EncryptionConfiguration>
        <ReplicaKmsKeyID>AWS KMS key IDs to use for encrypting object replicas</
ReplicaKmsKeyID>
      </EncryptionConfiguration>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

レプリケーション設定を *example-s3-source-bucket* バケットに追加するには、以下を実行します。

a. AWS CLI ではレプリケーション設定を JSON 形式で指定する必要があります。以下の JSON を、ローカルコンピュータの現在のディレクトリにあるファイル (replication.json) に保存します。

```
{
   "Role":"IAM-Role-ARN",
   "Rules":[
      {
         "Status":"Enabled",
         "Priority":1,
         "DeleteMarkerReplication":{
            "Status":"Disabled"
         },
         "Filter":{
            "Prefix":"Tax"
         },
         "Destination":{
            "Bucket":"arn:aws:s3:::example-s3-destination-bucket",
            "EncryptionConfiguration":{
               "ReplicaKmsKeyID":"AWS KMS key IDs (in ARN format) to use for
 encrypting object replicas"
            }
         },
         "SourceSelectionCriteria":{
            "SseKmsEncryptedObjects":{
               "Status":"Enabled"
            }
         }
      }
   ]
}
```

b. JSON を編集して、*example-s3-destination-bucket* バケット、*AWS KMS key IDs (in ARN format)*、および *IAM-role-ARN* の値を指定します。変更を保存します。

c. 次のコマンドを使用して、レプリケーション設定を *example-s3-source-bucket* バケットに追加します。必ず *example-s3-source-bucket* バケット名を指定してください。

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket example-s3-source-bucket \
--profile acctA
```

6. 設定をテストして、暗号化されたオブジェクトがレプリケートされることを確認します。Amazon S3 コンソールで、次の操作を行います。

   a. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
   b. *example-s3-source-bucket* バケットに Tax という名前のフォルダを作成します。
   c. サンプルオブジェクトをフォルダに追加します。必ず暗号化オプションを選択し、オブジェクトを暗号化するための KMS キーを指定してください。
   d. *example-s3-destination-bucket* バケットにオブジェクトのレプリカが含まれていることと、設定で指定した KMS キーを使用して暗号化されていることを確認します。詳細については、「the section called “レプリケーションステータスの取得”」を参照してください。

AWS SDK の使用

レプリケーション設定を追加するコード例については、「AWS SDK の使用」を参照してください。レプリケーション設定を適切に変更する必要があります。

概念については、「暗号化されたオブジェクトのレプリケート (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)」を参照してください。

Amazon S3 レプリカの変更同期によるメタデータ変更のレプリケート

Amazon S3 レプリカの変更を同期することで、タグ、ACL、オブジェクトロック設定などのオブジェクトメタデータをレプリカとレプリケート元オブジェクト間でレプリケートできます。デフォルトでは、Amazon S3 はレプリケート元オブジェクトからレプリカにのみメタデータをレプリケートします。レプリカの変更同期を有効にすると、Amazon S3 はレプリカコピーに加えられたメタデータの変更をレプリケート元オブジェクトにレプリケートし、レプリケーションを双方向にします。

レプリカ変更の同期の有効化

Amazon S3 レプリカの変更同期は、新規または既存のレプリケーションルールで使用できます。削除マーカーレプリケーションは、S3 バケット全体に適用することも、特定のプレフィックスを持つ Amazon S3 オブジェクトに適用することもできます。

Amazon S3 コンソールを使用してレプリカ変更の同期を有効にするには、ライブレプリケーションの設定例を参照してください。このトピックでは、バケットが同じまたは異なる AWS アカウントによって所有されている場合に、レプリケーション設定でレプリカ変更の同期を有効にする手順について説明します。

AWS Command Line Interface (AWS CLI) を使用してレプリカ変更の同期を有効にするには、ReplicaModifications が有効になっているレプリカを含むバケットにレプリケーション設定を追加する必要があります。双方向レプリケーションを設定するには、レプリケート元バケット (*example-s3-bucket1*) から、レプリカが含まれているバケット (*example-s3-bucket2*) に対するレプリケーションルールを作成します。次に、レプリカを含むバケット (*example-s3-bucket2*) からレプリケート元バケット (*example-s3-bucket1*) に対するレプリケーションルールを、もう 1 つ作成します。バケットは同一の、あるいは異なる AWS リージョン 内に置くことができます。

> **Note**
>
> オブジェクトのアクセスコントロールリスト (ACL)、オブジェクトのタグ、あるいはオブジェクトロックの設定など、レプリカメタデータの変更を複製するには、両方のバケットでレプリカ変更の同期を有効にする必要があります。これらのルールは、すべてのレプリケーションルールと同様に、 Amazon S3 バケット全体に適用することも、Amazon S3 オブジェクトのサブセットをプレフィックスまたはオブジェクトタグでフィルタリングして適用することも可能です。

以下の設定例では、Amazon S3 は *Tax* というプレフィックスのメタデータの変更を *example-s3-bucket* バケット (このバケットにはレプリケート元オブジェクトを含む) にレプリケートします。

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "SourceSelectionCriteria": {
                "ReplicaModifications":{
                    "Status": "Enabled"
                }
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
```

```
}
```

AWS CLI を使用してレプリケーションルールを作成する手順については、「同じアカウントが所有するレプリケート元バケットとレプリケート先バケットのレプリケーションの設定」を参照してください。

バケット間での削除マーカーのレプリケーション

デフォルトでは、S3 レプリケーションが有効で、レプリケーション元バケットでオブジェクトが削除されると、Amazon S3 はレプリケート元バケットにのみ削除マーカーを追加します。この動作で、悪意のある削除からデータを保護します。

削除マーカーレプリケーションが有効になっている場合、これらのマーカーはレプリケート先バケットにコピーされ、Amazon S3 はオブジェクトがレプリケート元バケットとレプリケート先バケットの両方で削除されたように動作します。削除マーカーの動作の詳細については、削除マーカーの使用を参照してください。

> **Note**
>
> 削除マーカーレプリケーションは、タグベースのレプリケーションルールではサポートされていません。削除マーカーレプリケーションは、S3 Replication Time Control の使用時に付与された 15 分間の SLA にも準拠しません。

レプリケーション設定の最新バージョンを使用していない場合、削除操作はレプリケーションに異なる影響を与えます。詳細については、「削除オペレーションがレプリケーションに与える影響」を参照してください。

削除マーカーレプリケーションの有効化

新規または既存のレプリケーションルールで、削除マーカーレプリケーションの使用を開始できます。削除マーカーレプリケーションは、S3 バケット全体に適用することも、特定のプレフィックスを持つ Amazon S3 オブジェクトに適用することもできます。

> **Note**
>
> 削除マーカーレプリケーションを有効化し、バケットに S3 ライフサイクルの有効期限ルールがある場合、S3 ライフサイクルの有効期限ルールによって追加された削除マーカーはレプリケート先バケットにレプリケートされません。

Amazon S3 コンソールを使用して削除マーカーレプリケーションを有効にするには、S3 コンソールの使用を参照してください。このトピックでは、バケットが同じまたは異なる AWS アカウントによって所有されている場合に、レプリケーション設定で削除マーカーレプリケーションを有効にする手順について説明します。

AWS Command Line Interface (AWS CLI) を使用して削除マーカーレプリケーションを有効にするには、DeleteMarkerReplication が有効になっているレプリケート元バケットにレプリケーション設定を追加する必要があります。

次の設定例では、*Tax* というプレフィクスの下にあるオブジェクトのレプリケート先バケットである *DOC-EXAMPLE-BUCKET* に、削除マーカーがレプリケートされます。

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
                "Status": "Enabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

AWS CLI を介してレプリケーションルールを作成する手順については、レプリケーションのチュートリアルセクションの「同じアカウントが所有するレプリケート元バケットとレプリケート先バケットのレプリケーションの設定」を参照してください。

ライブレプリケーションの管理または一時停止

ライブレプリケーションは、同一または異なる AWS リージョン にあるバケット間でオブジェクトを自動的に非同期でコピーする機能です。レプリケーションを設定すると、Amazon S3 は新しく作成されたオブジェクトと更新されたオブジェクトをレプリケート元バケットから指定された 1 つ以上のレプリケート先バケットにレプリケートします。

レプリケーションルールは、Amazon S3 コンソールを使用してレプリケート元のバケットに追加できます。レプリケーションルールにより、レプリケート元のソースバケットオブジェクトと、レプリケートされたオブジェクトが保存されるレプリケート先バケットが定義されます。レプリケーションの詳細については、「オブジェクトのレプリケーション」を参照してください。

[レプリケーション] ページで、レプリケーションルールを管理できます。レプリケーションルールは、追加、表示、有効化、無効化、削除できます。また、レプリケーションルールの優先度を変更することもできます。バケットへのレプリケーションルールの追加については、「S3 コンソールの使用」を参照してください。

Amazon S3 コンソールを使用して S3 バケットのレプリケーションルールを管理するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [汎用バケット] タブで、バケットの名前を選択します。
4. [管理] タブを選択して、[レプリケーションルール] まで下にスクロールします。
5. レプリケーションルールは、以下の方法で変更することができます。
   - レプリケーションルールを有効化または無効化するには、ルールの左側にあるオプションボタンを選択します。[アクション] メニューで、[ルールの有効化] または [ルールの無効化] を選択します。[アクション] メニューでは、バケットのすべてのルールを無効化、有効化、削除することもできます。

     > **Note**
     >
     > レプリケーションルールを無効化し、後でルールを再度有効化した場合、ルールの無効化中にレプリケートされなかった新規または変更されたオブジェクトは、ルールを再度有効にしても自動的にレプリケートされません。これらのオブジェクトをレプリケートするには、S3 バッチレプリケーションを使用する必要があります。詳細については、「the section called “既存のオブジェクトのレプリケーション”」を参照してください。

   - ルールの優先度を変更するには、ルールの左側にあるオプションボタンを選択して、[ルールの編集] を選択します。

     ルールの優先順位を指定して、複数のルールの範囲に含まれるオブジェクトによって引き起こされる競合を回避します。ルールが重複している場合、Amazon S3 はルールの優先度

を使用して適用するルールを決定します。数値が大きいほど、優先度が高くなります。ルーティングの優先度の詳細については、「レプリケーション設定」をご覧ください。

レプリケーションの一時停止または停止

レプリケーションを一時停止し、後で自動的に再開するには、AWS Fault Injection Service の `aws:s3:bucket-pause-replication` アクションを使用できます。詳細については、「AWS Fault Injection Service ユーザーガイド」の「aws:s3:bucket-pause-replication」および「Pause S3 Replication」を参照してください。

Amazon S3 でレプリケーションを停止するには、レプリケーションルールを無効化することをお勧めします。レプリケーションルールを無効化し、後でルールを再度有効化した場合、ルールの無効化中にレプリケートされなかった新規または変更されたオブジェクトは、ルールを再度有効にしても自動的にレプリケートされません。これらのオブジェクトをレプリケートするには、S3 バッチレプリケーションを使用する必要があります。詳細については、「the section called “既存のオブジェクトのレプリケーション”」を参照してください。

AWS Identity and Access Management (IAM) ロール、AWS Key Management Service (AWS KMS) アクセス許可、または必要なアクセス許可を Amazon S3 に付与するバケットポリシーアクセス許可を削除すると、レプリケーションは停止します。ただし、レプリケーションが失敗するため、これらのアプローチはお勧めしません。Amazon S3 は、影響を受けるオブジェクトのレプリケーションステータスを `FAILED` として報告します。アクセス許可が後で復元された場合、`FAILED` とマークされたオブジェクトは自動的にレプリケートされません。これらのオブジェクトをレプリケートするには、S3 バッチレプリケーションを使用する必要があります。

レプリケーションメトリクスと S3 イベント通知による、進捗状況のモニタリング

S3 レプリケーションメトリクス は、レプリケーション設定のレプリケーションルールの詳細なメトリクスを提供します。レプリケーションメトリクスを使って、保留中のバイト数、保留中のオペレーション、レプリケーションに失敗したオペレーション、レプリケーションのレイテンシーを追跡すると、レプリケーションの進行状況を 1 分単位でモニタリングできます。

S3 Replication Time Control (S3 RTC) を有効にすると、S3 レプリケーションメトリクスが自動的にオンになります。ルールを作成または編集するときに、S3 RTC とは独立して S3 レプリケーションメトリクスを有効にすることもできます。S3 RTC には、サービスレベルアグリーメント (SLA) やしきい値の欠落に関する通知など、その他の機能が含まれています。詳細については、

「S3 Replication Time Control (S3 RTC) を使用してコンプライアンス要件を満たす」を参照してください。

保留中のバイト数、保留中のオペレーション、およびレプリケーション待ち時間のメトリクスは、S3 クロスリージョンレプリケーション (S3 CRR) または S3 同一リージョンレプリケーション (S3 SRR) でレプリケートされた新しいオブジェクトにのみ適用されます。レプリケーションに失敗したオペレーションメトリクスは、S3 CRR または S3 SRR でレプリケートされた新しいオブジェクトと、S3 Batch レプリケーションでレプリケートされた既存のオブジェクトの両方を追跡します。設定の問題のトラブルシューティングを支援するために、Amazon S3 イベント通知を設定して、レプリケーションの失敗イベントを受信することもできます。

Amazon S3 イベント通知を有効にすると、S3 レプリケーションメトリクスは次のメトリクスを Amazon CloudWatch に発行します。

- レプリケーションを保留しているバイト – 特定のレプリケーションルールのレプリケーションを保留しているオブジェクトの合計バイト数。
- [レプリケーションレイテンシー] – 特定のレプリケーションルールで、レプリケート先バケットがレプリケート元バケットの背後にある最大秒数。
- レプリケーションを保留している操作 – 特定のレプリケーションルールでレプリケーションを保留している操作の数。このメトリクスは、オブジェクト、削除マーカー、タグ、アクセスコントロールリスト (ACL)、S3 オブジェクトロックに関連するオペレーションを追跡します。
- [レプリケーションに失敗したオペレーション] – 特定のレプリケーションルールでレプリケーションに失敗したオペレーションの数。このメトリクスは、オブジェクト、削除マーカー、タグ、ACL、S3 オブジェクトロックに関連するオペレーションを追跡します。他のレプリケーションメトリクスとは異なり、このメトリクスは S3 CRR または S3 SRR でレプリケートされる新しいオブジェクトと、S3 Batch レプリケーションでレプリケートされる既存のオブジェクトの両方に適用されます。

> **Note**
>
> [レプリケーションに失敗したオペレーション] は、S3 レプリケーションの失敗を 1 分間隔で集計して追跡します。レプリケーションに失敗した特定のオブジェクトとその失敗理由を特定するには、Amazon S3 イベント通知で `OperationFailedReplication` イベントをサブスクライブしてください。詳細については、「Amazon S3 イベント通知によるレプリケーション失敗イベントの受信」を参照してください。

ジョブがまったく実行されない場合、メトリクスは Amazon CloudWatch に送信されません。例えば、S3 Batch レプリケーションジョブを実行するために必要なアクセス許可がない場合、またはレプリケーション設定のタグまたはプレフィックスが一致しない場合、ジョブは実行されません。

トピック

S3 レプリケーションメトリクスの有効化

S3 レプリケーションメトリクスは、新規または既存のレプリケーションルールを使用して開始できます。レプリケーションルールを S3 バケット全体に適用するか、特定のプレフィックスやタグを持つ Amazon S3 オブジェクトに適用するかを選択できます。

このトピックでは、ソースバケットとレプリケーション先バケットが同じまたは異なる AWS アカウント で所有されている場合に、レプリケーション設定で S3 レプリケーションメトリクスを有効にする手順について説明します。

AWS Command Line Interface (AWS CLI) を使用してレプリケーションメトリクスを有効にするには、`Metrics` が有効になっているレプリケート元バケットにレプリケーション設定を追加する必要があります。この設定例では、*`Tax`* というプレフィックスの下にあるオブジェクトがレプリケート先バケットである *`DOC-EXAMPLE-BUCKET`* にレプリケートされて、それらのオブジェクトのメトリクスが生成されます。

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "Metrics": {
```

```
                    "Status": "Enabled"
                }
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

レプリケーションルールの作成方法の詳細については、「同じアカウントが所有するレプリケート元バケットとレプリケート先バケットのレプリケーションの設定」を参照してください。

S3 コンソールでのレプリケーションメトリクスの表示の詳細については、「Amazon S3 コンソールを使用したレプリケーションメトリクスの表示」を参照してください。

> **Note**
>
> S3 レプリケーションメトリクスは、Amazon CloudWatch カスタムメトリクスと同じ料金レートで請求されます。詳細については、「Amazon CloudWatch の料金」を参照してください。

Amazon S3 イベント通知によるレプリケーション失敗イベントの受信

S3 イベント通知は、オブジェクトがレプリケート先 AWS リージョン にレプリケートされない場合に通知できます。Amazon S3 イベントは、Amazon Simple Queue Service (Amazon SQS)、Amazon Simple Notification Service (Amazon SNS)、または AWS Lambda を通じて使用できます。詳細については、「the section called “Amazon S3 イベント通知”」を参照してください。

S3 イベント通知によってキャプチャされた失敗コードのリストについては、「Amazon S3 レプリケーションの失敗の理由」を参照してください。

S3 Storage Lens でのレプリケーションメトリクスの表示

レプリケーションルール数メトリクスを含む S3 レプリケーションの詳細なメトリクスを取得するには、Amazon S3 ストレージレンズを使用できます。S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージの分析機能です。詳細については、「Using S3 Storage Lens to protect your data」(S3 ストレージレンズを使用してデータを保護する) を参照してください。メトリクスの完全なリストについては、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

Amazon S3 コンソールを使用したレプリケーションメトリクスの表示

Amazon S3 の Amazon CloudWatch メトリクスには、ストレージメトリクス、リクエストメトリクス、レプリケーションメトリクスの 3 種類があります。AWS Management Console または Amazon S3 API を使用して S3 Replication Time Control (S3 RTC) でレプリケーションを有効にすると、S3 レプリケーションメトリクスが自動的にオンになります。ルールを作成または編集するときに、S3 RTC とは独立して S3 レプリケーションメトリクスを有効にすることもできます。

レプリケーションメトリクスは、レプリケーション設定のルール ID を追跡します。レプリケーションルール ID は、プレフィックス、タグ、またはその両方の組み合わせに固有です。

Amazon S3 の CloudWatch メトリクスの詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。

前提条件

S3 レプリケーションメトリクスを持つレプリケーションルールを有効にします。

レプリケーションメトリクスを表示するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。[バケット] リストで、レプリケーションメトリクスを取得するオブジェクトが含まれているバケットの名前を選択します。
3. [メトリクス] タブをクリックします。
4. [Replication metrics (レプリケーションメトリクス)] で、[Replication rules (レプリケーションルール)] を選択します。
5. [Display charts (チャートの表示)] を選択します。

   Amazon S3 には、[レプリケーションレイテンシー (秒単位)]、[レプリケーション保留中のバイト数]、[レプリケーションを保留中のオペをレーション]、および [レプリケーションに失敗したオペレーション] のグラフが表示されます。

これで、選択したルールの [レプリケーションのレイテンシー (秒)]、[レプリケーション保留中のオペレーション]、[レプリケーションを保留中のバイト数]、[レプリケーションに失敗したオペレーション] のレプリケーションメトリクスを表示できます。S3 レプリケーション時間コントロールを使用している場合、Amazon CloudWatch は、各レプリケーションルールで S3 RTC を有効にしてから 15 分後にレプリケーションメトリクスのレポートを開始します。Amazon S3 コンソールまた

は CloudWatch コンソールでレプリケーションメトリクスを表示できます。詳細については、「S3 RTC でのレプリケーションメトリクス」を参照してください。

> **Note**
>
> Amazon S3 ストレージレンズ を使用して Amazon S3 コンソールで S3 レプリケーションの詳細なメトリクスを表示することもできます。S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージの分析機能です。詳細については、「Using S3 Storage Lens to protect your data」(S3 ストレージレンズを使用してデータを保護する) を参照してください。メトリクスの完全なリストについては、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

Amazon S3 レプリケーションの失敗の理由

次の表は、Amazon S3 レプリケーションが失敗した理由を示しています。Amazon S3 イベント通知で FailureReason イベントを受け取ると、これらの理由を確認できます。S3 イベント通知は、Amazon Simple Queue Service (Amazon SQS)、Amazon Simple Notification Service (Amazon SNS)、または AWS Lambda を使用して受信できます。詳細については、「Amazon S3 イベント通知」を参照してください。

失敗の理由は、S3 バッチレプリケーション完了レポートでも確認できます。詳細については、「バッチレプリケーション完了レポート」を参照してください。

レプリケーションの失敗の理由	説明
`AssumeRoleNotPermitted`	Amazon S3 は、レプリケーション設定またはバッチオペレーションジョブで指定された AWS Identity and Access Management (IAM) ロールを引き受けることができません。
`DstBucketInvalidRegion`	宛先バケットがバッチオペレーションジョブで指定されたのと同じ AWS リージョンにありません。このエラーはバッチレプリケーションに固有のものです。

レプリケーションの失敗の理由	説明
DstBucketNotFound	Amazon S3 は、レプリケーション設定で指定されたレプリケート先バケットを見つけることができません。
DstBucketObjectLockConfigMissing	オブジェクトロックが有効になっているレプリケート元バケットからオブジェクトをレプリケートするには、レプリケート先バケットでもオブジェクトロックが有効になっている必要があります。このエラーは、レプリケート先バケットでオブジェクトロックが有効になっていない可能性があることを示しています。詳細については、「オブジェクトロックの考慮事項」を参照してください。
DstBucketUnversioned	S3 レプリケート先バケットでバージョニングが有効になっていません。S3 レプリケーションでオブジェクトをレプリケートするには、レプリケート先バケットのバージョニングを有効にします。
DstDelObjNotPermitted	Amazon S3 は、レプリケート先バケットに削除マーカーをレプリケートできません。レプリケート先バケットに対する s3:ReplicateDelete 権限がない可能性があります。
DstKmsKeyInvalidState	レプリケート先バケットの AWS Key Management Service (AWS KMS) キーが無効な状態です。必要な AWS KMS キーを確認して有効にします。AWS KMS キーの管理について詳しくは、AWS Key Management Service デベロッパーガイドの「AWS KMS キーのキーステータス」を参照してください。

レプリケーションの失敗の理由	説明
DstKmsKeyNotFound	レプリケーション設定でレプリケート先バケットに設定されている AWS KMS キーは存在しません。
DstMultipartCompleteNotPermitted	Amazon S3 は、レプリケート先バケットでオブジェクトのマルチパートアップロードを完了できません。レプリケート先バケットに対する s3:ReplicateObject 権限がない可能性があります。
DstMultipartInitNotPermitted	Amazon S3 は、レプリケート先バケットへのオブジェクトのマルチパートアップロードを開始できません。レプリケート先バケットに対する s3:ReplicateObject 権限がない可能性があります。
DstMultipartPartUploadNotPermitted	Amazon S3 は、レプリケート先バケットにマルチパートオブジェクトをアップロードできません。レプリケート先バケットに対する s3:ReplicateObject 権限がない可能性があります。
DstObjectHardDeleted	S3 バッチレプリケーションでは、レプリケート先バケットからオブジェクトのバージョン ID で削除されたオブジェクトの再レプリケーションはサポートされません。このエラーはバッチレプリケーションに固有のものです。
DstPutAclNotPermitted	Amazon S3 は、レプリケート先バケットにオブジェクトアクセスコントロールリスト (ACL) をレプリケートできません。レプリケート先バケットに対する s3:ReplicateObject 権限がない可能性があります。

レプリケーションの失敗の理由	説明
DstPutLegalHoldNotPermitted	イミュータブルなオブジェクトをレプリケートしている際は、Amazon S3 ではレプリケート先オブジェクトにオブジェクトロックのリーガルホールドを設定できません。レプリケート先バケットに対する s3:PutObjectLegalHold 権限がない可能性があります。詳細については、「リーガルホールド」を参照してください。
DstPutObjectNotPermitted	Amazon S3 は、レプリケート先バケットにオブジェクトをレプリケートできません。レプリケート先バケットに対する s3:ReplicateObject または s3:ObjectOwnerOverrideToBucketOwner 権限がない可能性があります。
DstPutTaggingNotPermitted	Amazon S3 は、レプリケート先バケットにオブジェクトタグをレプリケートできません。レプリケート先バケットに対する s3:ReplicateObject 権限がない可能性があります。
DstVersionNotFound	Amazon S3 は、メタデータをレプリケートする必要があるレプリケート先バケットで必要なオブジェクトバージョンを見つけることができません。
InitiateReplicationNotPermitted	Amazon S3 は、オブジェクトのレプリケーションを開始できません。バッチオペレーションジョブに対する s3:InitiateReplication 権限がない可能性があります。このエラーはバッチレプリケーションに固有のものです。

レプリケーションの失敗の理由	説明
SrcBucketInvalidRegion	レプリケート元バケットがバッチオペレーションジョブで指定されたのと同じ AWS リージョンにありません。このエラーはバッチレプリケーションに固有のものです。
SrcBucketNotFound	Amazon S3 はレプリケート元バケットを見つけることができません。
SrcBucketReplicationConfigMissing	Amazon S3 はソースバケットのレプリケーション設定を見つけることができませんでした。
SrcGetAclNotPermitted	Amazon S3 は、レプリケート元バケットにあるオブジェクトにアクセスしてレプリケートを行うことができません。レプリケート元バケットオブジェクトに対する s3:GetObjectVersionAcl 権限がない可能性があります。 ソースバケットのオブジェクトは、バケット所有者が所有している必要があります。ACL が有効になっている場合は、[オブジェクト所有権] が [希望するバケット所有者] または [オブジェクトライター] に設定されているかどうかを確認してください。オブジェクト所有権が [希望するバケット所有者] に設定されている場合、バケット所有者がオブジェクト所有者になるためには、ソースバケットオブジェクトに bucket-owner-full-control ACL が必要です。ソースアカウントは、オブジェクト所有権を [バケット所有者の強制] に設定して、ACL を無効にすることで、バケット内のすべてのオブジェクトの所有権を取得できます。

レプリケーションの失敗の理由	説明
SrcGetLegalHoldNotPermitted	Amazon S3 は S3 オブジェクトロックのリーガルホールド情報にアクセスできません。
SrcGetObjectNotPermitted	Amazon S3 は、レプリケート元バケットにあるオブジェクトにアクセスしてレプリケートを行うことができません。レプリケート元バケットに対する s3:GetObjectVersionForReplication 権限がない可能性があります。
SrcGetRetentionNotPermitted	Amazon S3 は S3 オブジェクトロックの保持期間情報にアクセスできません。
SrcGetTaggingNotPermitted	Amazon S3 はレプリケート元バケットのオブジェクトタグ情報にアクセスできません。レプリケート元バケットに対する s3:GetObjectVersionTagging 権限がない可能性があります。
SrcHeadObjectNotPermitted	Amazon S3 はレプリケート元バケットからオブジェクトメタデータを取得できません。レプリケート元バケットに対する s3:GetObjectVersionForReplication 権限がない可能性があります。
SrcKeyNotFound	Amazon S3 は、レプリケートするソースオブジェクトキーを見つけることができません。ソースオブジェクトはレプリケーションが完了する前に削除された可能性があります。

レプリケーションの失敗の理由	説明
SrcKmsKeyInvalidState	レプリケート元バケットの AWS KMS キーは有効な状態ではありません。必要な AWS KMS キーを確認して有効にします。AWS KMS キーの管理について詳しくは、AWS Key Management Service デベロッパーガイドの「AWS KMS キーのキーステータス」を参照してください。
SrcObjectNotEligible	一部のオブジェクトはレプリケーションの対象外です。これは、オブジェクトのストレージクラスが原因であるか、オブジェクトタグがレプリケーション構成と一致しないことが原因である可能性があります。
SrcObjectNotFound	ソースオブジェクトが存在しません。
SrcReplicationNotPending	Amazon S3 はすでにこのオブジェクトをレプリケートしています。このオブジェクトはもう保留中レプリケーションではなくなっています。
SrcVersionNotFound	Amazon S3 は、レプリケートするソースオブジェクトバージョンを見つけることができません。ソースオブジェクトバージョンはレプリケーションが完了する前に削除された可能性があります。

関連トピック

ライブレプリケーションのアクセス許可の設定

レプリケーションのトラブルシューティング

レプリケーションステータス情報の取得

レプリケーションステータスは、レプリケートされるオブジェクトの現在の状態を判断するのに役立ちます。レプリケート元オブジェクトのレプリケーションステータスは、 PENDING、COMPLETEDま

たはFAILEDのいずれかを返します。レプリカのレプリケーションステータスがREPLICAに返されます。

トピック

- レプリケーションステータスの概要
- 複数のレプリケート先バケットにレプリケートする場合のレプリケーションステータス
- Amazon S3 レプリカ変更の同期が有効になっている場合のレプリケーションステータス
- レプリケーションステータスの検索

レプリケーションステータスの概要

レプリケーションには、レプリケーションを設定するレプリケート元バケットと、Amazon S3 がオブジェクトをレプリケートするレプリケート先バケットがあります。これらのバケットからオブジェクト (GET オブジェクトを使用) またはオブジェクトメタデータ (HEAD オブジェクトを使用) をリクエストすると、Amazon S3 はレスポンスとして`x-amz-replication-status` ヘッダーを返します。

- レプリケート元バケットのオブジェクトをリクエストする場合、リクエストしたオブジェクトがレプリケーション対象であると、Amazon S3 は `x-amz-replication-status` ヘッダーを返します。

  たとえば、レプリケーション設定でオブジェクトプレフィックス `TaxDocs` を指定して、キー名のプレフィックス `TaxDocs` が付いたオブジェクトのみをレプリケートするように Amazon S3 に指示しているとします。このキー名のプレフィックスを持つ、アップロードしたすべてのオブジェクト (`TaxDocs/document1.pdf` など) がレプリケートされます。このキー名のプレフィックスが付いたオブジェクトのリクエストでは、Amazon S3 が、オブジェクトのレプリケーション状態が `PENDING`、`COMPLETED`、または `FAILED` の値のいずれかの `x-amz-replication-status` ヘッダーを返します。

  > **Note**
  >
  > オブジェクトをアップロードした後で、オブジェクトのレプリケーションに失敗した場合、レプリケーションを再試行できません。もう一度オブジェクトをアップロードする必要があります。レプリケーションロールの許可、AWS KMS 許可、またはバケットの許可がないなどの問題がある場合、オブジェクトは `FAILED` の状態に移行します。バケットやリージョンが使用できないなどの一時的な障害が発生した場合、レプリケーションのス

テータスは FAILED にはならず、PENDING のままになります。リソースがオンラインに戻ると、S3 はこれらのオブジェクトのレプリケーションを再開します。

- レプリケート先バケットからオブジェクトをリクエストした場合、リクエストされたオブジェクトが Amazon S3 によって作成されたレプリカであるときに、Amazon S3 は値がREPLICAである`x-amz-replication-status`ヘッダーを返します。

> **Note**
>
> レプリケーションが有効になっているレプリケート元バケットからオブジェクトを削除する前に、削除する前にオブジェクトのレプリケーションステータスをチェックして、そのオブジェクトがレプリケートされていることを確認します。
> レプリケート元バケットでライフサイクル設定が有効である場合、Amazon S3 は、オブジェクトステータスがCOMPLETEDまたはFAILEDと表示されるまでライフサイクルアクションを停止します。

複数のレプリケート先バケットにレプリケートする場合のレプリケーションステータス

オブジェクトを複数のレプリケート先バケットにレプリケートした場合、`x-amz-replication-status`ヘッダーの動作はそれぞれ異なります。レプリケート元オブジェクトのヘッダーは、すべてのレプリケート先へのレプリケーションが成功した場合のみ、COMPLETEDの値を返します。すべてのレプリケート先に対してレプリケーションが完了するまで、ヘッダーはPENDINGの値のままになります。1 つ以上のレプリケート先がレプリケーションに失敗すると、ヘッダーはFAILEDを返します。

Amazon S3 レプリカ変更の同期が有効になっている場合のレプリケーションステータス

レプリケーションルールが Amazon S3 レプリカの変更を有効にすると、レプリカは REPLICA 以外のステータスをレポートできます。メタデータの変更がレプリケート中の場合は、`x-amz-replication-status` ヘッダーは PENDING を返します。レプリカ変更の同期がメタデータのレプリケートに失敗した場合、ヘッダーは FAILED を返します。メタデータが正しくレプリケートされると、レプリカはヘッダー REPLICA を返します。

レプリケーションステータスの検索

バケット内のオブジェクトのレプリケーションステータスを取得するには、Amazon S3 インベントリツールを使用できます。Amazon S3 は、インベントリ設定で指定したレプリケート先バケットに CSV ファイルを送信します。Amazon Athena を使用して、インベントリレポートのレプリケー

ションステータスをクエリすることもできます。Amazon S3 インベントリのさらなる詳細については、Amazon S3 インベントリ を参照してください。

オブジェクトのレプリケーションステータスは、コンソール、AWS Command Line Interface (AWS CLI)、または AWS SDK で確認できます。

S3 コンソールの使用

S3 コンソールで、[オブジェクト管理の概要] の、オブジェクトの [詳細] ページで、オブジェクトのレプリケーションステータスを確認できます。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、バケット名を選択します。
3. [オブジェクト] リストで、オブジェクト名を選択します。
4. [Properties] (プロパティ) タブで [Object management overview] (オブジェクト管理の概要) を見つけると、[Replication status] (レプリケーションステータス) を確認できます。

AWS CLI の使用

次のように、head-objectコマンドを使用してオブジェクトメタデータを取得します。

```
aws s3api head-object --bucket source-bucket --key object-key --version-id object-version-id
```

このコマンドは、以下のレスポンス例に示すように、ReplicationStatus を含むオブジェクトメタデータを返します。

```
{
   "AcceptRanges":"bytes",
   "ContentType":"image/jpeg",
   "LastModified":"Mon, 23 Mar 2015 21:02:29 GMT",
   "ContentLength":3191,
   "ReplicationStatus":"COMPLETED",
   "VersionId":"jfnW.HIMOfYiD_9rGbSkmroXsFj3fqZ.",
   "ETag":"\"6805f2cfc46c0f04559748bb039d69ae\"",
   "Metadata":{

   }
}
```

AWS SDK の使用

次のコードフラグメントは、それぞれ AWS SDK for Java および AWS SDK for .NET のレプリケーションステータスを取得します。

Java

```
GetObjectMetadataRequest metadataRequest = new GetObjectMetadataRequest(bucketName,
 key);
ObjectMetadata metadata = s3Client.getObjectMetadata(metadataRequest);

System.out.println("Replication Status : " +
 metadata.getRawMetadataValue(Headers.OBJECT_REPLICATION_STATUS));
```

.NET

```
GetObjectMetadataRequest getmetadataRequest = new GetObjectMetadataRequest
    {
         BucketName = sourceBucket,
         Key        = objectKey
    };

GetObjectMetadataResponse getmetadataResponse =
 client.GetObjectMetadata(getmetadataRequest);
Console.WriteLine("Object replication status: {0}",
 getmetadataResponse.ReplicationStatus);
```

S3 バッチレプリケーションを使用した既存のオブジェクトのレプリケーション

S3 バッチレプリケーションを使用すると、次のタイプのオブジェクトをレプリケートできます。

- レプリケーション設定が実行される前に存在していたオブジェクト
- 以前にレプリケートされたオブジェクト
- レプリケーションに失敗したオブジェクト

バッチオペレーションジョブを使用して、これらのオブジェクトをオンデマンドでレプリケートできます。S3 バッチレプリケーションは、Amazon S3 バケット間で新しいオブジェクトを継続的かつ自動的にレプリケートするライブレプリケーションとは異なります。

バッチレプリケーションを開始するには、次の操作を行います。

- 新しいレプリケーションルールまたはレプリケート先のバッチレプリケーションを開始する — 1 回限りのバッチレプリケーションジョブは、新しいレプリケーション設定で最初のルールを作成する場合や、Amazon S3 を介して既存の設定に新しいレプリケート先を追加する場合に作成できます。
- 既存のレプリケーション設定のバッチレプリケーションを開始する — S3 バッチオペレーションを使用して、Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 REST API を介し、新しいバッチレプリケーションジョブを作成できます。

バッチレプリケーションジョブが終了すると、完了レポートが表示されます。レポートを使用してジョブを検査する方法の詳細については、「ジョブステータスと完了レポートの追跡」を参照してください。

S3 バッチレプリケーションに関する考慮事項

- レプリケート元バケットには既存のレプリケーション設定が必要です。レプリケーションを有効にするには、「ライブレプリケーションの設定」と「ライブレプリケーションの設定例」を参照してください。
- バケットで S3 ライフサイクルが設定されている場合は、バッチレプリケーションジョブがアクティブな際には、ライフサイクルルールを無効にすることをお勧めします。これは、レプリケート元バケットとレプリケート先バケットの間で等価性を確保するのに役立ちます。これを行わない場合、これらのバケットが分岐し、レプリケート先バケットがレプリケート元バケットの完全なレプリカではなくなる可能性があります。たとえば、次のシナリオを考えてみます。
  - レプリケート元バケットには、オブジェクトとそのオブジェクトの削除マーカーの複数のバージョンがあります。
  - レプリケート元とレプリケート先のバケットには、期限切れの削除マーカーを削除するライフサイクル設定が設定されています。

  このシナリオでは、バッチレプリケーションにより、オブジェクトのバージョンがレプリケートされる前に、レプリケート先バケットに削除マーカーがレプリケートされることがあります。その場合、オブジェクトのバージョンがレプリケートされる前に、ライフサイクル設定で削除マーカーの有効期限が切れ、削除マーカーがレプリケート先バケットから削除される可能性があります。
- バッチオペレーションジョブを実行するために指定する AWS Identity and Access Management IAM ロールには、基になるバッチレプリケーションを実行する権限が必要です。IAM ロールの作

成の詳細については、「バッチレプリケーション用の IAM ポリシーの設定」を参照してください。

- バッチレプリケーションには、Amazon S3 で生成できるマニフェストが必要です。生成されたマニフェストは、レプリケーション元バケットと同じ AWS リージョンに保存されている必要があります。マニフェストを生成しない場合は、レプリケートするオブジェクトを含む Amazon S3 インベントリレポートまたは CSV ファイルを指定できます。
- バッチレプリケーションでは、レプリケート先バケットからオブジェクトのバージョン ID で削除されたオブジェクトの再レプリケーションはサポートされません。これらのオブジェクトを再レプリケートするには、バッチコピージョブを使用してソースオブジェクトを所定の場所でコピーします。これらのオブジェクトを所定の場所でコピーすると、レプリケート元バケットにオブジェクトの新しいバージョンが作成され、レプリケート先バケットへのレプリケーションが自動的に開始されます。レプリケート先バケットを削除して再作成しても、レプリケーションは開始されません。

  バッチコピーの詳細については、「バッチ操作を使用してオブジェクトをコピーする例」を参照してください。
- S3 バケットでレプリケーションルールを使用している場合は、レプリケーション設定を更新して、レプリケーションルールにアタッチされている IAM ロールに、オブジェクトをレプリケートするための適切な権限を付与してください。この IAM ロールには、ソースバケットとレプリケート先バケットの両方でレプリケーションを実行する権限が必要です。
- 短期間に同じバケットに対して複数の Batch レプリケーションジョブを送信すると、Amazon S3 はそれらのジョブを同時に実行します。
- 2 つの異なるバケットに対して複数の Batch レプリケーションジョブを送信する場合、Amazon S3 がすべてのジョブを同時に実行するわけではないことに注意してください。アカウントで一度に実行できる Batch レプリケーションジョブの数を超えると、Amazon S3 は優先度の低いジョブを一時停止して優先度の高いジョブを処理します。優先度の高い項目が完了すると、一時停止していたジョブは再びアクティブになります。
- バッチオペレーションは、S3 Glacier Flexible Retrieval ストレージクラスと S3 Glacier Deep Archive ストレージクラスに保存されたオブジェクトに対してはサポートされていません。
- アーカイブアクセスまたはディープアーカイブアクセスのストレージ階層に保存されている S3 Intelligent-Tiering オブジェクトのバッチレプリケーションを行うには、まず復元リクエストを開始し、オブジェクトが高頻度アクセス階層に移動されるまで待つ必要があります。

バッチレプリケーションジョブのマニフェストの指定

マニフェストは、Amazon S3 が動作するオブジェクトキーを含む Amazon S3 オブジェクトです。バッチレプリケーションジョブを作成する場合は、ユーザー生成のマニフェストを指定するか、レプリケーション設定に基づいて Amazon S3 でマニフェストを生成させる必要があります。

ユーザーが作成したマニフェストを提供する場合は、Amazon S3 インベントリレポートまたは CSV ファイルの形式である必要があります。マニフェスト内のオブジェクトがバージョニング対応のバケット内にある場合は、そのオブジェクトのバージョン ID を指定する必要があります。マニフェストで指定されたバージョン ID を持つオブジェクトのみがレプリケートされます。マニフェストの指定の詳細については、「マニフェストの指定」を参照してください。

Amazon S3 がユーザーに代わってマニフェストファイルを生成するように選択した場合、リストされるオブジェクトでは、ソースバケットのすべてのレプリケーション設定と同じレプリケーション元のバケット、プレフィックス、タグを使用します。生成されたマニフェストでは、Amazon S3 はオブジェクトの適格なバージョンをすべてレプリケートします。

> **Note**
>
> Amazon S3 によるマニフェストの生成を選択する場合は、マニフェストがレプリケーション元バケットと同じ AWS リージョンに保存されている必要があります。

バッチレプリケーションジョブのフィルター

バッチレプリケーションジョブを作成するときに、オプションで、オブジェクトの作成日やレプリケーションステータスなどの追加のフィルターを指定して、ジョブのスコープを縮小できます。

オブジェクトのレプリケーションは、`ObjectReplicationStatuses` 値を基に、以下の値を 1 つ以上指定することでフィルターできます。

- `"NONE"` - Simple Storage Service (Amazon S3) がこれまでオブジェクトのレプリケートを試みたことがないことを示します。
- `"FAILED"` - Amazon S3 が以前にオブジェクトのレプリケートを試みたがオブジェクトのレプリケーションに失敗したことを示します。
- `"COMPLETED"` - Simple Storage Service (Amazon S3) が以前にオブジェクトを正常にレプリケートしたことを示します。
- `"REPLICA"` - Amazon S3 が別のソースからレプリケートしたレプリカオブジェクトであることを示します。

レプリケーションステータスの詳細については、「レプリケーションステータス情報の取得」を参照してください。

バッチレプリケーションジョブをフィルターしない場合、バッチオペレーションは、デフォルトでレプリケートされない特定のオブジェクトを除き、レプリケーション設定のルールに一致するマニフェスト内のすべてのオブジェクト (ObjectReplicationStatus に関係なく) のレプリケーションを試みます。詳細については、「the section called “レプリケーション設定でレプリケートされないものは何ですか?”」を参照してください。

目標によっては、以下の 1 つまたは複数の値に ObjectReplicationStatuses を設定する場合があります。

- レプリケートされたことがない既存のオブジェクトのみをレプリケートするには、"NONE" のみを含めます。
- 以前にレプリケートに失敗したオブジェクトのレプリケートのみを再試行するには、"FAILED" のみを含めます。
- 既存のオブジェクトをレプリケートし、以前にレプリケートに失敗したオブジェクトのレプリケーションを再試行するには、"NONE" と "FAILED" の両方を含めます。
- 別のレプリケート先にレプリケートされたオブジェクトをレプリケート先バケットにバックフィルするには、"COMPLETED" を含めます。
- 以前にレプリケートオブジェクトをレプリケートするには、"REPLICA" を含めます。

バッチレプリケーション完了レポート

バッチレプリケーションジョブを作成するときに、CSV 完了レポートをリクエストできます。このレポートには、オブジェクト、レプリケーションの成功コードまたは失敗コード、出力、および説明が表示されます。ジョブの追跡と完了レポートの追跡については、「完了レポート」を参照してください。

レプリケーション失敗コードのリストと説明については、「Amazon S3 レプリケーションの失敗の理由」を参照してください。

バッチレプリケーションのトラブルシューティングについては、「バッチレプリケーションエラー」を参照してください。

バッチレプリケーションの使用開始

バッチレプリケーションの使用方法の詳細については、「チュートリアル: S3 バッチレプリケーションによる Amazon S3 バケット内の既存のオブジェクトのレプリケーション」を参照してください。

バッチレプリケーション用の IAM ポリシーの設定

S3 バッチレプリケーションはバッチオペレーションジョブの一種であるため、バッチオペレーション AWS Identity and Access Management(IAM) ロールを作成し、ユーザーに代わってアクションを実行するための Amazon S3 許可を付与する必要があります。また、バッチレプリケーション IAM ポリシーをバッチオペレーション IAM ロールにアタッチする必要があります。次の例では、バッチレプリケーションジョブを開始するバッチオペレーションの許可を与える IAM ロールを作成します。

IAM ロールとポリシーを作成する

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. [Access management] (アクセス管理) で、[Roles] (ロール) を選択します。
3. [ロールの作成] を選択します。
4. 信頼されるエンティティのタイプとして AWS のサービス を、サービスとして Amazon S3 を、ユースケースとして [S3 Batch Operations] (S3 バッチオペレーション) を選択します。
5. [次へ: アクセス許可] を選択します。
6. [ポリシーの作成] を選択します。
7. [JSON] を選択し、マニフェストに基づいて次のいずれかのポリシーを挿入します。

> **Note**
>
> マニフェストを生成する場合、またはマニフェストを提供する場合は、異なる許可が必要です。詳細については、「バッチレプリケーションジョブのマニフェストの指定」を参照してください。

S3 で生成されたマニフェストを使用して保存する場合のポリシー

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
         ]
      },
      {
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:PutInventoryConfiguration"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***"
         ]
      },
      {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::*** completion report bucket ****/*",
            "arn:aws:s3:::*** manifest bucket ****/*"
         ]
      }
   ]
}
```

ユーザー指定のマニフェストを使用する場合のポリシー

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
         ]
      },
            {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::*** completion report bucket ****/*"
         ]
      }
   ]
}
```

8. [次へ: タグ] を選択します。
9. [次へ: レビュー] を選択します。
10. ポリシーの名前を指定し、[Create policy] (ポリシーを作成) を選択します。
11. このポリシーをロールに添付し、[Next: Tags] (次へ:タグ) を選択します。
12. [次へ: レビュー] を選択します。

13. ロールの名前を指定し、[Create role] (ロールを作成) を選択します。

信頼ポリシーを確認する

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. [Access management] (アクセス管理) で、[Roles] (ロール) を選択し、新しく作成したロールを選択します。
3. [Trust relationships] (信頼関係) タブで、[Edit trust relationship] (信頼関係を編集) を選択します。
4. このロールが次の信頼ポリシーを使用していることを確認します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

最初のレプリケーションルールまたは新しいレプリケート先にバッチレプリケーションジョブを作成する

新しいレプリケーション設定で最初のルールを作成する場合、または AWS Management Consoleを介して新しいレプリケート先を既存の設定に追加する場合は、オプションでバッチレプリケーションジョブを作成できます。

既存の設定に対して、新しいレプリケート先を追加せずにバッチレプリケーションを使用するには、「既存のレプリケーションルールのバッチレプリケーションジョブを作成する」を参照してください。

AWS Management Console を介して新しいレプリケーションルールまたはレプリケート先にバッチレプリケーションを使用する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets] (バケット) リストで、レプリケートするオブジェクトが含まれるバケットの名前を選択します。
3. 新しいレプリケーションルールを作成するか、既存のルールを編集するには、[管理] を選択し、下へスクロールして [レプリケーションルール] を選択します。

   - 新しいレプリケーションルールを作成するには、[レプリケーションルールの作成] を選択します。

     > **Note**
     >
     > ベーシックレプリケーションルールのセットアップ方法の例については、「ライブレプリケーションの設定例」を参照してください。

   - 既存のバックアップルールを編集するには、ルールを選択してから、[編集] を選択します。
4. 新しいレプリケーションルールを作成するか、既存のレプリケーションルールのレプリケーション先を編集して、[保存] を選択します。

   新しいレプリケーション設定で最初のルールを作成した後、または既存の設定を編集して新しいレプリケート先を追加すると、[Replicate existing objects?] (既存のオブジェクトをレプリケーションしますか?) ダイアログが表示され、バッチレプリケーションジョブを作成するオプションが表示されます。
5. このジョブを今すぐ実行する場合は、[はい、既存のオブジェクトをレプリケートする] を選択します。

   または、[いいえ、既存のオブジェクトをレプリケートしない] を選択すると、このジョブは後ほど実行できます。
6. S3 Batch レプリケーションジョブを作成します。S3 バッチレプリケーションジョブには、いくつかの設定があります。

   ジョブ実行オプション

   S3 Batch レプリケーションジョブをすぐに実行する場合は、[Job runs automatically when ready] (準備ができたらジョブは自動的に実行) を選択できます。後でジョブを実行する場

合は、[Job waits to be run when ready] (ジョブは準備ができるまで実行を待機) を選択します。

[Job runs automatically when ready] (ジョブは準備ができしだい自動で実行) を選択すると、バッチオペレーションマニフェストを作成および保存することはできません。バッチオペレーションマニフェストを保存するには、[Job waits to be run when ready] (ジョブは準備ができたら実行を待機) を選択します。

バッチオペレーションマニフェスト

マニフェストとは、指定されたアクションを実行するすべてのオブジェクトのリストです。バッチオペレーションマニフェストを保存することもできます。S3 インベントリファイルと同様に、マニフェストは CSV ファイルとして保存され、バケットに保存されます。バッチオペレーションマニフェストの詳細については、「マニフェストの指定」を参照してください。

完了レポート

S3 バッチオペレーションは、マニフェストで指定された各オブジェクトに対して 1 つのタスクを実行します。完了レポートは、追加の設定を必要としない統合形式でタスクの結果を表示する簡単な方法です。すべてのタスクまたは失敗したタスクについてのみ完了レポートをリクエストできます。完了レポートの詳細については、「完了レポート」を参照してください。

アクセス許可

レプリケーション失敗の最も一般的な原因の 1 つは、提供されている AWS Identity and Access Management (IAM) ロールの権限が不十分であることです。このロールの作成の詳細については、「バッチレプリケーション用の IAM ポリシーの設定」を参照してください。

7. [Create Batch Operations job] (バッチオペレーションジョブを作成) を選択します。

既存のレプリケーションルールのバッチレプリケーションジョブを作成する

AWS SDK、AWS Command Line Interface (AWS CLI)、Simple Storage Service (Amazon S3) コンソールを使用して、既存のレプリケーション設定に S3 バッチレプリケーションを設定できます。バッチレプリケーションの概要については、「S3 バッチレプリケーションを使用した既存のオブジェクトのレプリケーション」を参照してください。

前提条件として、ユーザーに代わりアクションを実行するための Simple Storage Service (Amazon S3) の許可を付与するバッチオペレーション AWS Identity and Access Management (IAM) ロールを作成する必要があります。「バッチレプリケーション用の IAM ポリシーの設定」を参照してください。

バッチレプリケーションジョブが終了すると、完了レポートが表示されます。レポートを使用してジョブを検査する方法の詳細については、「ジョブステータスと完了レポートの追跡」を参照してください。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. Amazon S3 コンソールのナビゲーションペインで [Batch Operations (バッチ操作)] を選択します。
3. [ジョブの作成] を選択します。
4. ジョブを作成する [Region (リージョン)] を選択します。
5. [Manifest format] (マニフェスト形式) を選択します。この例では、既存の S3 レプリケーション設定に基づいてマニフェストを作成する方法を説明します。

   > **Note**
   >
   > マニフェストとは、指定されたアクションを実行するすべてのオブジェクトのリストです。バッチオペレーションマニフェストの詳細については、「マニフェストの指定」を参照してください。マニフェストの準備ができている場合は、S3 インベントリレポート manifest.jsonまたはCSV。マニフェスト内のオブジェクトがバージョニング対応のバケット内にある場合は、そのオブジェクトのバージョン ID を指定する必要があります。マニフェストの作成の詳細については、「マニフェストの指定」を参照してください。

6. レプリケーション構成に基づいてマニフェストを作成するには、[Create manifest using S3 Replication configuration] (S3 レプリケーション設定を使用してマニフェストを作成する) を選択します。レプリケーション設定をレプリケート元バケットに追加します。
7. (オプション) オブジェクトの作成日やレプリケーションステータスなどの追加のフィルターを含めることができます。レプリケーションステータスによるフィルター方法の例については、「バッチレプリケーションジョブのマニフェストの指定」を参照してください。

8. マニフェストを保存するには、[Save Batch Operations manifest] (バッチオペレーションマニフェストを保存) を選択します。

   a. マニフェストを生成して保存する場合は、[Bucket in this account] (このアカウントのバケット) または [Bucket in another AWS アカウント] (別の AWS アカウント のバケット) のいずれかを選択する必要があります。テキストボックスにバケット名を指定します。

      > **Note**
      >
      > 生成されたマニフェストは、レプリケーション元バケットと同じ AWS リージョンに保存されている必要があります。

   b. [暗号化タイプ] を選択します。

9. (オプション) [Description] (説明) に入力します。

10. 必要に応じて、ジョブの [Priority] (優先度) を調整します。番号が高いほど、優先順位が高いことを表します。Simple Storage Service (Amazon S3) は、優先度の低いジョブの前に優先度の高いジョブを実行しようとします。ジョブの優先度の詳細については、「ジョブの優先度の割り当て」を参照してください。

11. (オプション) 完了レポートを生成します。生成するには [Generate completion report] (完了レポートを生成) を選択します。

    完了レポートを生成する場合は、[Failed tasks only] (失敗したタスクのみ) または [All tasks] (すべてのタスク) をレポートするかを選択肢、レポートのレプリケート先バケットを指定する必要があります。

12. 有効な IAM ロールを選択します。

    > **Note**
    >
    > IAM ロールの作成の詳細については、「バッチレプリケーション用の IAM ポリシーの設定」を参照してください。

13. (オプション) バッチレプリケーションジョブにジョブタグを追加します。

14. [次へ] を選択します。

15. 設定を確認し、[Create job] (ジョブを作成) を選択します。

S3 マニフェストとともに、AWS CLI を使用する

次の例では、S3 が AWS アカウント *111122223333* に生成したマニフェストを使用して S3 バッチレプリケーションジョブを作成します。この例では、既存のオブジェクトと以前にレプリケートに失敗したオブジェクトの複製を試みます。レプリケーションステータスによるフィルタリングの詳細については、「バッチレプリケーションジョブのマニフェストの指定」を参照してください。

```
aws s3control create-job --account-id 111122223333 --operation
 '{"S3ReplicateObject":{}}' --report '{"Bucket":"arn:aws:s3:::***
 completion report bucket ****","Prefix":"batch-replication-report",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}'
 --manifest-generator '{"S3JobManifestGenerator": {"ExpectedBucketOwner":
 "111122223333", "SourceBucket": "arn:aws:s3:::*** replication source bucket
 ***", "EnableManifestOutput": false, "Filter": {"EligibleForReplication": true,
 "ObjectReplicationStatuses": ["NONE","FAILED"]}}}' --priority 1 --role-arn
 arn:aws:iam::111122223333:role/batch-Replication-IAM-policy --no-confirmation-required
 --region source-bucket-region
```

> **Note**
>
> ジョブは、同じ AWS リージョン レプリケーション元バケットから開始する必要があります。IAM ロール `role/batch-Replication-IAM-policy` は以前に作成されました。「バッチレプリケーション用の IAM ポリシーの設定」を参照してください。

バッチレプリケーションジョブを正常に開始すると、応答としてジョブ ID を受け取ります。次のコマンドを使用して、このジョブをモニタリングできます。

```
aws s3control describe-job --account-id 111122223333 --job-id job-id --region source-
bucket-region
```

ユーザー提供のマニフェストとともに AWS CLI を使用する

次の例では、AWS アカウント *111122223333* のユーザー定義のマニフェストを使用して S3 バッチレプリケーションジョブを作成します。マニフェスト内のオブジェクトがバージョニング対応のバケット内にある場合は、そのオブジェクトのバージョン ID を指定する必要があります。マニフェストで指定されたバージョン ID を持つオブジェクトのみがレプリケートされます。マニフェストの作成の詳細については、「マニフェストの指定」を参照してください。

```
aws s3control create-job --account-id 111122223333 --operation
 '{"S3ReplicateObject":{}}' --report '{"Bucket":"arn:aws:s3:::***
 completion report bucket ****","Prefix":"batch-replication-report",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}'
 --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key","VersionId"]},"Location":{"ObjectArn":"arn:aws:s3:::*** completion
 report bucket ****/manifest.csv","ETag":"Manifest Etag"}}' --priority 1 --role-arn
 arn:aws:iam::111122223333:role/batch-Replication-IAM-policy --no-confirmation-required
 --region source-bucket-region
```

> **Note**
>
> ジョブは、同じ AWS リージョン レプリケーション元バケットから開始する必要があります。IAM ロール role/*batch-Replication-IAM-policy* は以前に作成されました。「バッチレプリケーション用の IAM ポリシーの設定」を参照してください。

バッチレプリケーションジョブを正常に開始すると、応答としてジョブ ID を受け取ります。次のコマンドを使用して、このジョブをモニタリングできます。

```
aws s3control describe-job --account-id 111122223333 --job-id job-id --region source-
bucket-region
```

タグを使用してストレージを分類する

オブジェクトのタグ付けを使用してストレージを分類します。各タグはキーと値のペアです。

新しいオブジェクトのアップロード時のタグ付け、または既存のオブジェクトのタグ付けを行うことができます

- 1 つのオブジェクトに最大 10 個のタグを関連付けることができます。オブジェクトに関連付けるタグには一意のタグキーが必要です。
- タグキーには最大 128 個の Unicode 文字、タグ値には最大 256 個の Unicode 文字を使用できます。Amazon S3 オブジェクトタグは、内部的に UTF-16 で表現されています。UTF-16 では、文字は 1 文字または 2 文字の位置を消費することに注意してください。
- キーと値は大文字と小文字が区別されます。

- タグの制限の詳細については、「AWS Billing and Cost Management ユーザーガイド」の「User-defined tag restrictions」を参照してください。基本的なタグの制限については、「Amazon EC2 ユーザーガイド」の「タグの制限」を参照してください。

例

次のタグ付けの例について説明します。

Example PHI 情報

保護されるべき医療情報 (PHI) データがオブジェクトに含まれているとします。このオブジェクトは次に示すキーと値のペアを使用してタグ付けできます。

```
PHI=True
```

または

```
Classification=PHI
```

Example プロジェクトのファイル

プロジェクトファイルを S3 バケットに保存するとします。このようなオブジェクトは、`Project` というキーと値を使用して次のようにタグ付けします。

```
Project=Blue
```

Example 複数のタグ

次に示すように複数のタグを 1 つのオブジェクトに付けることができます。

```
Project=x
Classification=confidential
```

キー名のプレフィックスとタグ

オブジェクトキー名のプレフィックスにより、ストレージを分類することもできます。ただし、プレフィックスベースの分類は 1 次元的です。次のオブジェクトキー名について説明します。

```
photos/photo1.jpg
```

```
project/projectx/document.pdf
project/projecty/document2.pdf
```

これらのキー名には、プレフィックスとして photos/、project/projectx/、および project/projecty/ が付いています。このプレフィックスによって 1 つのディメンションでの分類が行われます。つまり、1 つのプレフィックスのオブジェクトすべてが 1 つのカテゴリになります。たとえば、プレフィックス project/projectx はプロジェクト x に関連するすべてのドキュメントを識別します。

タグ付けによってもう 1 つのディメンションを設定できます。プロジェクト x カテゴリ内に photo1 が必要な場合、そのようにオブジェクトにタグ付けできます。

追加の利点

データの分類だけでなく、タグ付けには以下に示すような利点もあります。

- オブジェクトのタグを使用するとアクセス許可をきめ細かく制御できます。例えば、特定のタグが付けられた読み取り専用オブジェクトへのアクセス許可をユーザーに付与できます。
- オブジェクトのタグを使用すると、ライフサイクルルールでキー名プレフィックスに加えてタグに基づくフィルタを指定でき、オブジェクトライフサイクルをきめ細かく管理できます。
- Amazon S3 分析を使用する場合は、フィルターを設定することで、分析対象のオブジェクトをオブジェクトタグまたはキー名のプレフィックスに基づいてグループ化するか、プレフィックスとタグの両方に基づいてグループ化することができます。
- また、Amazon CloudWatch メトリクスをカスタマイズして、特定のタグフィルターごとに情報を表示することもできます。詳細については次のセクションで説明します。

> **Important**
>
> 機密データ (個人を特定できる情報 (PII) または保護すべき医療情報 (PHI) を含むオブジェクトのラベルとしてタグを使用できます。しかし、タグそのものに機密情報を含めることはできません。

1 つのリクエストで複数の Amazon S3 オブジェクトにオブジェクトタグセットを追加する

1 つのリクエストでオブジェクトタグセットを複数の Amazon S3 オブジェクトに追加するには、S3 バッチオペレーションを使用できます。S3 バッチオペレーションには、オペレーション対象のオブ

ジェクトのリストを指定します。S3 バッチオペレーションは、各 API オペレーションを呼び出して、指定されたオペレーションを実行します。1 つのバッチオペレーションジョブで、エクサバイトのデータを含む数十億ものオブジェクトに対して、指定されたオペレーションを実行できます。

S3 バッチオペレーション機能は、進捗状況の追跡、通知の送信、すべてのアクションの詳細な完了レポートの保存を行い、フルマネージドの監査可能なサーバーレスエクスペリエンスを提供します。S3 バッチオペレーションは、Amazon S3 コンソール、AWS CLI、AWS SDK、または REST API を通じて使用できます。詳細については、「the section called “バッチ操作の基本”」を参照してください。

オブジェクトタグの詳細については、オブジェクトタグの管理を参照してください。

オブジェクトのタグ付けに関連する API オペレーション

Amazon S3 では、特にオブジェクトのタグ付けについて次の API オペレーションがサポートされています。

オブジェクトの API オペレーション

- PUT Object tagging – オブジェクトのタグを置換します。リクエストボディにタグを指定します。この API を使用したオブジェクトタグ管理の 2 つのシナリオは次のとおりです。
  - オブジェクトにタグがない – この API を使用して一連のタグをオブジェクトに付けることができます (オブジェクトには事前にタグが付いていない)。
  - オブジェクトに既存のタグセットがある – 既存のタグセットを変更するには、最初に既存のタグセットを取得してクライアント側で変更してから、この API を使用してタグセットを置換する必要があります。

    > **Note**
    >
    > タグセットを空にしてこのリクエストを送信すると、Amazon S3 によってオブジェクトの既存のタグセットが削除されます。この方法を使用すると、範囲 1 リクエスト (PUT) に対して料金が発生します。詳細については、Amazon S3 の料金 を参照してください。
    >
    > 料金が発生することなく同じ結果が得られるため、DELETE Object tagging リクエストが推奨されます。

- GET Object tagging – オブジェクトに関連付けられているタグセットを返します。Amazon S3 によってレスポンス本文でオブジェクトタグが返されます。

- DELETE Object tagging – オブジェクトに関連付けられているタグセットを削除します。

タグ付けをサポートする他の API オペレーション

- PUT Object および Initiate Multipart Upload – オブジェクトを作成するときにタグを指定できます。`x-amz-tagging` リクエストヘッダーを使用してタグを指定します。
- GET Object – Amazon S3 はタグセットを返さずに、`x-amz-tag-count` ヘッダーでオブジェクトのタグ数を返します (リクエスタにタグの読み取りアクセス許可がある場合のみ)。これは、ヘッダーのレスポンスサイズが 8 KB に制限されるためです。タグを表示するには、GET Object tagging API オペレーションで別のリクエストを作成します。
- POST Object – POST リクエストでタグを指定できます。

  リクエスト内のタグが 8 KB の HTTP リクエストヘッダーサイズ制限を超えない場合は、PUT `Object` API を使用してタグ付きのオブジェクトを作成できます。指定するタグがヘッダーサイズ制限を超える場合は、この POST メソッドを使用して本文にタグを指定できます。

  PUT Object - Copy – リクエストに `x-amz-tagging-directive` を指定し、タグをコピー (デフォルト動作) するか、リクエストに指定された新しいタグセットでタグを置換するように Amazon S3 に指示します。

次の点に注意してください。

- S3 オブジェクトのタグ付けには強固な整合性があります。詳細については、「Amazon S3 のデータ整合性モデル」を参照してください。

追加の設定

このセクションでは、オブジェクトのタグ付けが他の設定にどのように関連するかを説明します。

オブジェクトのタグ付けとライフサイクル管理

バケットライフサイクル設定では、ルールを適用するオブジェクトのサブセットを選択するためにフィルタを指定できます。キー名プレフィックスまたはオブジェクトタグ (あるいは両方) に基づいてフィルタを指定できます。

Amazon S3 バケットに写真 (RAW 形式および完成形式) を保存するとします。このようなオブジェクトには次のようにタグ付けできます。

```
phototype=raw
or
phototype=finished
```

写真の作成後、しばらくしてから RAW 写真を S3 Glacier にアーカイブすることを検討する場合があります。特定のタグ (photos/) を持つキー名プレフィックス (phototype=raw) でオブジェクトのサブセットを特定するフィルタを使用して、ライフサイクルルールを設定できます。

詳細については、「ストレージのライフサイクルの管理」を参照してください。

オブジェクトのタグ付けとレプリケーション

レプリケーションをバケットに設定した場合、タグを読み取るアクセス許可を Amazon S3 に付与すると、Amazon S3 によってタグがレプリケートされます。詳細については、「ライブレプリケーションの設定」を参照してください。

オブジェクトタグ付けイベント通知

オブジェクトタグがオブジェクトに追加または削除されたときに通知を受け取るように、Amazon S3 イベント通知を設定できます。`s3:ObjectTagging:Put` イベントタイプは、タグがオブジェクト上に置かれたとき、または既存のタグが更新されたときに通知します。`s3:ObjectTagging:Delete` イベントタイプは、タグがオブジェクトから削除されたときに通知します。詳細については、イベント通知を有効 を参照してください。

オブジェクトのタグ付けの詳細については、次のトピックを参照してください。

トピック

- タグ付けとアクセスコントロールポリシー
- オブジェクトタグの管理

タグ付けとアクセスコントロールポリシー

アクセス許可ポリシー (バケットポリシーとユーザーポリシー) を使用して、オブジェクトのタグ付けに関連するアクセス許可を管理することもできます。ポリシーアクションについては次のトピックを参照してください。

- オブジェクト操作
- バケットオペレーション

オブジェクトのタグを使用して、アクセス許可を管理するためのきめ細かいアクセスコントロールが可能になります。オブジェクトタグに基づいて条件付きアクセス許可を与えることができます。Amazon S3 でサポートされる次の条件キーを使用すると、オブジェクトタグに基づいて条件付きアクセス許可を付与できます。

- `s3:ExistingObjectTag/<tag-key>` – この条件キーを使用して、既存のオブジェクトタグに特定のタグキーと値があることを確認します。

  > **Note**
  >
  > PUT Object オペレーションおよび DELETE Object オペレーションのアクセス許可を与える場合、この条件キーはサポートされません。つまり、オブジェクトをその既存のタグに基づいて削除または上書きするアクセス許可を、ユーザーに許可または拒否するポリシーは作成できません。

- `s3:RequestObjectTagKeys` – この条件キーを使用して、オブジェクトに指定できるタグキーを制限します。これが役立つのは、PutObjectTagging、PutObject および POST Object リクエストを使用してオブジェクトにタグを付けるときです。
- `s3:RequestObjectTag/<tag-key>` – この条件キーを使用して、オブジェクトに指定できるタグキーと値を制限します。これが役立つのは、PutObjectTagging、PutObject、および POST Bucket リクエストを使用してオブジェクトにタグを付けるときです。

Amazon S3 サービス固有の条件キーの詳細なリストについては、「条件キーを使用したバケットポリシーの例」を参照してください。次のアクセス許可ポリシーを使用して、オブジェクトのタグ付けによってきめ細かいアクセス許可管理がどのように実現するかを説明します。

Example 1: 特定のタグキーと値を持つオブジェクトの読み取りのみをユーザーに許可する

以下のアクセス許可ポリシーでは、environment: production タグキーと値を持つオブジェクトのみを読み取れるように制限しています。このポリシーは s3:ExistingObjectTag 条件キーを使用してタグキーと値を指定します。

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Principal": {
      "AWS": [
        "arn:aws:iam::111122223333:role/JohnDoe"
```

```
      ]
    },
    "Effect": "Allow",
    "Action": ["s3:GetObject", "s3:GetObjectVersion"],
    "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
    "Condition": {
      "StringEquals":
        {"s3:ExistingObjectTag/environment": "production"}
    }
  }
  ]
}
```

Example 2: ユーザーが追加できるオブジェクトタグキーを制限する

次のアクセス許可ポリシーは、s3:PutObjectTagging アクションを実行する権限をユーザーに許可します (ユーザーが既存のオブジェクトにタグを付けることができます)。この条件は s3:RequestObjectTagKeys 条件キーを使用して、Owner や CreationDate などの許可されたタグキーを指定します。詳細については、「IAM ユーザーガイド」の「複数のキーの値をテストする条件の作成」を参照してください。

このポリシーは、リクエストで指定されたすべてのタグキーが承認されたタグキーであることを保証します。条件の ForAnyValue 修飾子によって、指定したキーの少なくとも 1 つがリクエストに存在することが保証されます。

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
            "arn:aws:iam::111122223333:role/JohnDoe"
         ]
       },
 "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"ForAnyValue:StringEquals": {"s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
```

```
          ]
        }
      }
    }
  ]
}
```

Example 3: ユーザーにオブジェクトタグの追加を許可する場合は特定のタグキーと値が必要

次のポリシーの例では、s3:PutObjectTagging アクションを実行するアクセス許可をユーザーに付与します (ユーザーが既存のオブジェクトにタグを追加することができます)。この条件により、値が *X* に設定された特定のタグキー (*Project* など) をユーザーが含めることが求められます。

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
       "arn:aws:iam::111122223333:user/JohnDoe"
        ]
       },
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"StringEquals": {"s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

オブジェクトタグの管理

このセクションでは、AWS SDK for Java および .NET、または Amazon S3 コンソールを使用してオブジェクトタグを管理する方法について説明します。

オブジェクトのタグ付けにより、ストレージを分類する方法が提供されます。各タグは、以下のルールに準拠したキーと値のペアです。

- 1 つのオブジェクトに最大 10 個のタグを関連付けることができます。オブジェクトに関連付けるタグには一意のタグキーが必要です。
- タグキーには最大 128 個の Unicode 文字、タグ値には最大 256 個の Unicode 文字を使用できます。Amazon S3 オブジェクトタグは、内部的に UTF-16 で表現されています。UTF-16 では、文字は 1 文字または 2 文字の位置を消費することに注意してください。
- キーと値は大文字と小文字が区別されます。

オブジェクトタグの詳細については、タグを使用してストレージを分類するを参照してください。タグの制限の詳細については、AWS Billing and Cost Management ユーザーガイドのユーザー定義タグの制限を参照してください。

S3 コンソールの使用

オブジェクトにタグを追加するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、タグを追加するオブジェクトが含まれるバケットの名前を選択します。

   必要に応じてフォルダに移動することもできます。
3. [オブジェクト] リストで、タグを追加するオブジェクトの名前の横にあるチェックボックスをオンにします。
4. [Actions] メニューで、[Edit tags] を選択します。
5. リストされたオブジェクトを確認し、[タグの追加] を選択します。
6. 各オブジェクトタグはキーと値のペアです。キーと値を入力します。別のタグを追加するには、[タグの追加] を選択します。

   オブジェクトには最大 10 個のタグを入力できます。
7. [Save changes] (変更を保存) をクリックします。

   Amazon S3 が指定されたオブジェクトにタグを追加します。

詳細については、このガイドの「Amazon S3 コンソールでのオブジェクトのプロパティの表示」および「オブジェクトのアップロード」も参照してください。

AWS SDK の使用

Java

次の例では、AWS SDK for Java を使用して新しいオブジェクトのタグを設定し、既存のオブジェクトのタグを取得または置き換える方法を示します。オブジェクトのタグ付けの詳細については、「タグを使用してストレージを分類する」を参照してください。作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

public class ManagingObjectTags {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Object key ***";
        String filePath = "*** File path ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Create an object, add two new tags, and upload the object to Amazon
 S3.
            PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName,
 new File(filePath));
            List<Tag> tags = new ArrayList<Tag>();
            tags.add(new Tag("Tag 1", "This is tag 1"));
```

```
            tags.add(new Tag("Tag 2", "This is tag 2"));
            putRequest.setTagging(new ObjectTagging(tags));
            PutObjectResult putResult = s3Client.putObject(putRequest);

            // Retrieve the object's tags.
            GetObjectTaggingRequest getTaggingRequest = new
 GetObjectTaggingRequest(bucketName, keyName);
            GetObjectTaggingResult getTagsResult =
 s3Client.getObjectTagging(getTaggingRequest);

            // Replace the object's tags with two new tags.
            List<Tag> newTags = new ArrayList<Tag>();
            newTags.add(new Tag("Tag 3", "This is tag 3"));
            newTags.add(new Tag("Tag 4", "This is tag 4"));
            s3Client.setObjectTagging(new SetObjectTaggingRequest(bucketName,
 keyName, new ObjectTagging(newTags)));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

次の例では、AWS SDK for .NET タグを使用して新しいオブジェクトのタグを設定し、既存のオブジェクトのタグを取得または置き換える方法を示します。オブジェクトのタグ付けの詳細については、「タグを使用してストレージを分類する」を参照してください。

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
```

```
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    public class ObjectTagsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for the new object ***";
        private const string filePath = @"*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            PutObjectWithTagsTestAsync().Wait();
        }

        static async Task PutObjectWithTagsTestAsync()
        {
            try
            {
                // 1. Put an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    FilePath = filePath,
                    TagSet = new List<Tag>{
                        new Tag { Key = "Keyx1", Value = "Value1"},
                        new Tag { Key = "Keyx2", Value = "Value2" }
                    }
                };

                PutObjectResponse response = await
 client.PutObjectAsync(putRequest);
                // 2. Retrieve the object's tags.
                GetObjectTaggingRequest getTagsRequest = new GetObjectTaggingRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };
```

```
                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);
                for (int i = 0; i < objectTags.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}",
objectTags.Tagging[i].Key, objectTags.Tagging[i].Value);


                // 3. Replace the tagset.

                Tagging newTagSet = new Tagging();
                newTagSet.TagSet = new List<Tag>{
                    new Tag { Key = "Key3", Value = "Value3"},
                    new Tag { Key = "Key4", Value = "Value4" }
                };


                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    Tagging = newTagSet
                };
                PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

                // 4. Retrieve the object's tags.
                GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest();
                getTagsRequest2.BucketName = bucketName;
                getTagsRequest2.Key = keyName;
                GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);
                for (int i = 0; i < objectTags2.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}",
objectTags2.Tagging[i].Key, objectTags2.Tagging[i].Value);

            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an
object"
```

```
                        , e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Encountered an error. Message:'{0}' when writing an object"
                    , e.Message);
            }
        }
    }
}
```

S3 バケットタグでのコスト配分タグの使用

個々のプロジェクトまたはプロジェクトのグループのストレージコストや他の基準を追跡するには、コスト配分タグを使用して Amazon S3 バケットにラベルを付けます。コスト割り当てタグはキーと値のペアで、ユーザーが S3 バケットに関連付けます。コスト配分タグを有効にすると、AWS はコスト配分レポートでリソースのコストの整理にタグを使用します。コスト割り当てタグは、バケットにラベルを付けるためだけに使用できます。オブジェクトにラベルを付けるために使用されるタグについては、「タグを使用してストレージを分類する」を参照してください。

コスト配分レポートには、アカウントの AWS 使用状況が製品カテゴリ別およびリンクされたアカウントユーザー別に表示されます。レポートには、同じ明細項目が詳細な請求レポートとして含まれ (「Amazon S3 の AWS 請求および使用状況レポートを理解する」を参照)、タグキーは追加列に含まれます。

AWS には、AWS が生成するタグとユーザー定義タグの 2 種類のコスト配分タグが用意されています。AWS は、Amazon S3 の CreateBucket イベントの後に、AWS が生成する `createdBy` タグを定義、作成、適用します。ユーザーは S3 バケットのユーザー定義タグを、定義、作成、適用します。

請求レポートに表示するには、請求およびコスト管理コンソールで両方の種類のタグを個別に有効化する必要があります。AWS が生成するタグの詳細については、「AWS 生成コスト配分タグ」を参照してください。

- コンソールでタグを作成するときは、「S3 バケットのプロパティを表示するには」を参照してください。
- Amazon S3 API を使用してタグを作成するには、Amazon Simple Storage Service API リファレンスの「PUT バケットタグ付け」を参照してください。

- AWS CLI を使用してタグを作成するには、AWS CLI コマンドリファレンスの「put-bucket-tagging」を参照してください。
- タグの有効化の詳細については、AWS Billing ユーザーガイドのコスト配分タグの使用を参照してください。

ユーザー定義のコスト配分タグ

ユーザー定義コスト配分タグには以下のコンポーネントがあります。

- タグキー。タグキーは、タグ名です。たとえば、project/Trinity タグでは project がキーです。タグキーは大文字小文字を区別する文字列で、1 ~ 128 個の Unicode 文字を含めることができます。

- タグ値。タグ値は必須文字列です。たとえば、project/Trinity タグでは Trinity が値です。タグ値は大文字小文字を区別する文字列で、0 ~ 256 個の Unicode 文字を含めることができます。

ユーザー定義タグで使用できる文字や他の制限の詳細については、AWS Billing ユーザーガイドのユーザー定義タグの制限を参照してください。ユーザー定義タグの詳細については、AWS Billing ユーザーガイドのユーザー定義のコスト配分タグを参照してください。

S3 バケットのタグ

各 S3 バケットはタグセットがあります。タグセットには、そのバケットに割り当てられているすべてのタグが含まれています。タグセットには最大 50 個のタグを含めることができ、空にすることもできます。キーはタグセット内で一意にする必要がありますが、タグセット内の値は一意である必要はありません。たとえば、同じ値を持つタグ project/Trinity と cost-center/Trinity を、同じタグセット内に持つことができます。

既存のタグと同じキーを持つタグをバケットに追加した場合、古い値は新しい値によって上書きされます。

AWS はタグに意味論的意味を適用しません。単なる文字列としてタグを解釈します。

タグを追加、一覧表示、編集、削除するには、Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、または Amazon S3 API を使用します。

詳細

- AWS Billing ユーザーガイドの「コスト配分タグの使用」
- Amazon S3 の AWS 請求および使用状況レポートを理解する

- Amazon S3 の AWS Billing レポート

Amazon S3 の請求および使用状況レポート

> **Important**
>
> 2024 年 5 月 13 日に、バケット所有者によって開始されていない不正なリクエストに対する料金を排除する変更のデプロイを開始しました。この変更のデプロイが完了すると、バケット所有者は、個々の AWS アカウントまたは AWS 組織の外部からリクエストが開始されたときに `AccessDenied` (HTTP `403 Forbidden`) エラーを返すリクエストに対してリクエストまたは帯域幅料金を請求されることはありません。請求されない HTTP `3XX` および `4XX` ステータスコードの全一覧については、「Amazon S3 のエラーレスポンスに対する請求」を参照してください。この請求の変更は、アプリケーションの更新を必要とせず、すべての S3 バケットに適用されます。すべての AWS リージョンでこの変更のデプロイが完了したら、ドキュメントを更新します。

Amazon S3 を使用する場合、前払い金を支払ったり、コンテンツの保存量をコミットしたりする必要はありません。他の AWS のサービス と同様、使用したときに使用した分のみのお支払いとなります。

AWS では、Amazon S3 について次のレポートを提供します。

- 請求レポート – Amazon S3 を含む、使用している AWS のサービス のすべてのアクティビティの概要ビューを提供する複数のレポート。AWS は、バケットがリクエスタ支払いバケットとして作成された場合を除き、常に S3 バケットの所有者に Amazon S3 料金を請求します。リクエスタ支払いの詳細については、「ストレージ転送と使用量のリクエスタ支払いバケットの使用」を参照してください。請求レポートの詳細については、「Amazon S3 の AWS Billing レポート」を参照してください。
- 使用状況レポート – 特定のサービスのアクティビティを時間、日、または月単位で集計した概要。含める使用タイプ、オペレーションを選択できます。また、データの収集方法も選択できます。詳細については、「AWSAmazon S3 の 使用状況レポート」を参照してください。

以下のトピックでは、Amazon S3 の請求と使用状況レポートについての情報を説明しています。

トピック

- AWSAmazon S3 の 使用状況レポート
- Amazon S3 の AWS 請求および使用状況レポートを理解する
- Amazon S3 のエラーレスポンスに対する請求

Amazon S3 の AWS Billing レポート

AWS からの毎月の請求書では、使用状況の情報とコストが AWS のサービス と機能ごとに分けられています。月別レポート、コスト割り当てレポート、詳細な請求レポートなど、利用できる複数の AWS Billing レポートがあります。請求レポートを表示する方法については、AWS Billing ユーザーガイドの「請求の表示」を参照してください。

AWS の使用状況を追跡し、アカウントに関連する推定請求額を示すよう AWS Cost and Usage Reports をセットアップできます。詳細については、「AWS Data Exports ガイド」の「AWSCost and Usage Reports とは」を参照してください。

Amazon S3 ストレージ使用状況に関する、請求レポートよりもさらに詳細な使用状況レポートをダウンロードできます。詳細については、「AWSAmazon S3 の 使用状況レポート」を参照してください。

次の表は、Amazon S3 の使用に関連する料金をまとめたものです。

Amazon S3 の利用料金

料金	コメント
ストレージ	S3 バケットにオブジェクトを保存すると料金が発生します。請求される料金は、オブジェクトのサイズ、その月の間のオブジェクトの保存期間、およびそれらのストレージクラスによって異なります。Amazon S3 には、S3 Standard、S3 Express One Zone、S3 Intelligent-Tiering、S3 Standard-IA (IA は少頻度アクセスを意味します)、S3 One Zone-IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive、または 低冗長化ストレージ (RRS) のストレージクラスが用意されています。ストレージクラスの詳細

料金	コメント
	については、「Amazon S3 ストレージクラスを使用する」を参照してください。 S3 バージョニングが有効になっている場合、保持されているオブジェクトのバージョンごとに課金されることに注意してください。バージョニングの詳細については、「S3 バージョニングの仕組み」を参照してください。
モニタリングと自動化	アクセスパターンをモニタリングし、S3 Intelligent-Tiering のアクセス層間でオブジェクトを移動するには、S3 Intelligent-Tiering ストレージクラスに保管されているオブジェクトごとにモニタリングとオートメーションの月額料金をお支払いいただきます。
リクエスト	また、S3 バケットやオブジェクトに対して行われた GET リクエストなど、リクエストに対しても料金が発生します。これには、ライフサイクル リクエストが含まれます。リクエストのレートは、リクエストの種類によって異なります。リクエストの料金の詳細については、「Amazon S3 の料金」を参照してください。
取得	S3 標準 – IA、S3 1 ゾーン – IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval、およびS3 Glacier Deep Archiveストレージに保存されているオブジェクトを取り出す時に料金が発生します。
早期削除	最小ストレージコミットメントを満たす前に S3 標準 – IA、S3 1 ゾーン – IA、S3 Glacier Flexible Retrieval、または S3 Glacier Deep Archive storage に保存されたオブジェクトを削除する場合、そのオブジェクトの早期削除料金が発生します。

料金	コメント
ストレージ管理	アカウントのバケットで有効になっているストレージ管理機能 (Amazon S3 インベントリ、分析、オブジェクトのタグ付け) に対してお支払いいただきます。
[帯域幅]	次の場合を除き、Amazon S3 に出入りするすべての帯域幅に対してお支払いいただきます。 • インターネットからのデータ転送 • インスタンスが S3 バケットと同じ AWS リージョン にある場合、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに転送されたデータ。 • Amazon CloudFront (CloudFront) に転送されたデータ。 また、Amazon S3 Transfer Acceleration を使用して転送されたデータに対しても料金をお支払いいただきます。

Amazon S3 のストレージ、データ転送、サービスの利用料金の詳細は、「Amazon S3 の料金」および「Amazon S3 のよくある質問」を参照してください。

Amazon S3 の請求および使用状況レポートで使用されるコードと略語については、「Amazon S3 の AWS 請求および使用状況レポートを理解する」を参照してください。

詳細情報

- AWSAmazon S3 の 使用状況レポート
- S3 バケットタグでのコスト配分タグの使用
- AWS Billing とコスト管理
- Amazon S3 の料金

AWSAmazon S3 の 使用状況レポート

使用状況レポートをダウンロードする際、使用状況を時間、日、または月単位で集計できます。Amazon S3 使用状況レポートでは、使用タイプおよび AWS リージョン 別にオペレーションが一覧表示されます。Amazon S3 ストレージの使用状況の詳細なレポートについては、動的に生成された AWS の使用状況レポートをダウンロードしてください。含める使用タイプ、オペレーション、期間を選択できます。また、データの収集方法も選択できます。使用状況レポートの詳細については、「AWS Data Exports ユーザーガイド」の「AWS 使用状況レポート」を参照してください。

Amazon S3 使用状況レポートには次の情報が含まれます。

- サービス – Amazon S3
- オペレーション – バケットまたはオブジェクトに対して実行されたオペレーション。Amazon S3 オペレーションの詳細な説明については、「使用状況レポートの追跡オペレーション」を参照してください。
- 使用状況 – 以下の値のいずれかです。
  - ストレージのタイプを識別するコード
  - リクエストのタイプを識別するコード
  - 取得のタイプを識別するコード
  - データ転送のタイプを識別するコード
  - S3 Intelligent-Tiering、S3 標準 – IA、S3 1 ゾーン低頻度アクセス (S3 1 ゾーン – IA)、S3 One Zone-Infrequent Access (S3 One Zone-IA)、S3 Glacier Flexible Retrieval、または S3 Glacier Deep Archive ストレージからの早期削除を識別するコード
  - `StorageObjectCount` – 特定のバケット内に保存されたオブジェクトの数

  Amazon S3 使用タイプの詳細な説明については、「Amazon S3 の AWS 請求および使用状況レポートを理解する」を参照してください。
- Resource – リストされた使用状況に関連付けられたバケットの名前。
- StartTime – 使用状況が適用される協定世界時 (UTC) による日付の開始時間。
- EndTime – 使用状況が適用がされる協定世界時 (UTC) による日付の終了時間。
- UsageValue – 以下のボリューム値のいずれかです。データの典型的な測定単位はギガバイト (GB) です。ただし、サービスとレポートによっては、テラバイト (TB) が代わりに表示されることがあります。
  - 指定した期間のリクエストの数
  - データ転送量

- 指定された時間内に格納されたデータの量
- S3 標準 – IA、または S3 1 ゾーン – IA、S3 Glacier Flexible Retrieval、または S3 Glacier Deep Archive ストレージからの復元に関連付けられたデータの量 (バイト単位)

> **Tip**
>
> オブジェクトに関して Amazon S3 が受信するすべてのリクエストに関する詳細な情報は、バケットのサーバーアクセスログ記録をオンにすると取得できます。詳細については、「サーバーアクセスログによるリクエストのログ記録」を参照してください。

使用状況レポートは、XML またはコンマ区切り値 (CSV) ファイルでダウンロードできます。スプレッドシートアプリケーションで開いた CSV 使用状況レポートの例を次に示します。

Service	Operation	UsageType	Resource	StartTime	EndTime	UsageValue
AmazonS3	HeadBucket	USW2-C3DataTransfer-Out-Bytes	admin-created3	6/1/2017 0:00	7/1/2017 0:00	15309
AmazonS3	PutObject	USW2-C3DataTransfer-In-Bytes	admin-created3	6/1/2017 0:00	7/1/2017 0:00	19062
AmazonS3	HeadBucket	USW2-Requests-Tier2	admin-created3	6/1/2017 0:00	7/1/2017 0:00	68
AmazonS3	PutObjectForRepl	USW1-Requests-SIA-Tier1	ca-example-bucket	6/1/2017 0:00	7/1/2017 0:00	178294
AmazonS3	PutObjectForRepl	USW1-USW2-AWS-In-Bytes	ca-example-bucket	6/1/2017 0:00	7/1/2017 0:00	387929083
AmazonS3	GetObjectForRepl	USW2-Requests-NoCharge	admin-created3	6/1/2017 0:00	7/1/2017 0:00	108
AmazonS3	GetObjectForRepl	USW2-USW1-AWS-Out-Bytes	my-test-bucket-bash	6/1/2017 0:00	7/1/2017 0:00	387910021

詳細については、「Amazon S3 の AWS 請求および使用状況レポートを理解する」を参照してください。

AWS 使用状況レポートのダウンロード

使用状況レポートは XML または CSV ファイルでダウンロードできます。

使用状況レポートのダウンロード

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. タイトルバーで、ユーザー名またはアカウント ID を選択し、[請求とコスト管理] を選択します。
3. ナビゲーションペインで、[コストと使用状況レポート] を選択します。
4. [AWS 使用状況レポート] で、[使用状況レポートの作成] を選択します。
5. [使用状況レポートのダウンロード] ページで、次の設定を選択します。

- [サービス] – [Amazon Simple Storage Service] を選択します。
- [Usage Types] (使用タイプ) – Amazon S3 の使用タイプの詳細については、Amazon S3 の AWS 請求および使用状況レポートを理解する を参照してください。
- [Operation] (オペレーション) – Amazon S3 オペレーションの詳細については、使用状況レポートの追跡オペレーション を参照してください。
- [Time Period] (期間) – レポートの対象とする期間。
- [Report Granularity (レポートの詳細度)] – 時間、日、月単位の小計をレポートに含めるかどうか。

6. [ダウンロード] を選択し、ダウンロード形式 (XML レポートまたは CSV レポート) を選択し、プロンプトに従ってレポートを開くか保存します。

詳細情報

- Amazon S3 の AWS 請求および使用状況レポートを理解する
- Amazon S3 の AWS Billing レポート

Amazon S3 の AWS 請求および使用状況レポートを理解する

> **Important**
>
> 2024 年 5 月 13 日に、バケット所有者によって開始されていない不正なリクエストに対する料金を排除する変更のデプロイを開始しました。この変更のデプロイが完了すると、バケット所有者は、個々の AWS アカウントまたは AWS 組織の外部からリクエストが開始されたときに `AccessDenied` (HTTP `403 Forbidden`) エラーを返すリクエストに対してリクエストまたは帯域幅料金を請求されることはありません。請求されない HTTP `3XX` および `4XX` ステータスコードの全一覧については、「Amazon S3 のエラーレスポンスに対する請求」を参照してください。この請求の変更は、アプリケーションの更新を必要とせず、すべての S3 バケットに適用されます。すべての AWS リージョンでこの変更のデプロイが完了したら、ドキュメントを更新します。

Amazon S3 請求および使用状況レポートではコードと略名を使用します。次に示す表の使用法タイプでは、*`region`*、*`region1`*、*`region2`* をこのリストの略語に置き換えます。

- APE1: アジアパシフィック (香港)

- APN1: アジアパシフィック (東京)
- APN2: アジアパシフィック (ソウル)
- APN3: アジアパシフィック (大阪)
- APS1: アジアパシフィック (シンガポール)
- APS2: アジアパシフィック (シドニー)
- APS3: アジアパシフィック (ムンバイ)
- APS4: アジアパシフィック (ジャカルタ)
- APS5: アジアパシフィック (ハイデラバード)
- APS6: アジアパシフィック (メルボルン)
- CAN1: カナダ (中部)
- CAN2: カナダ西部 (カルガリー)
- CNN1: 中国 (北京)
- CNW1: 中国 (寧夏)
- AFS1: アフリカ (ケープタウン)
- EUC2: 欧州 (チューリッヒ)
- EUN1: 欧州 (ストックホルム)
- EUS2: 欧州 (スペイン)
- EUC1: 欧州 (フランクフルト)
- EU: 欧州 (アイルランド)
- EUS1: 欧州 (ミラノ)
- EUW2: 欧州 (ロンドン)
- EUW3: 欧州 (パリ)
- ILC1: イスラエル (テルアビブ)
- MEC1: 中東 (アラブ首長国連邦)
- MES1: 中東 (バーレーン)
- SAE1: 南米 (サンパウロ)
- UGW1: AWS GovCloud (米国西部)
- UGE1: AWS GovCloud (米国東部)
- USE1 (またはプレフィックスなし): 米国東部 (バージニア北部)
- USE2: 米国東部 (オハイオ)

- USW1: 米国西部 (北カリフォルニア)
- USW2: 米国西部 (オレゴン)

次の表に示す S3 のマルチリージョンアクセスポイントの使用法タイプでは、*regiongroup1* と *regiongroup2* をこのリストの略語に置き換えます。

- AP: アジア太平洋
- AU: オーストラリア
- EU: ヨーロッパ
- IN: インド
- NA: 北米
- SA: 南米

リージョングループは、複数の AWS リージョン を地理的にグループ化したものです。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。AWS リージョン 別の料金については、「Amazon S3 の料金」を参照してください。

次の表の 1 列目は、請求および使用状況レポートに記載される使用タイプです。データの典型的な測定単位はギガバイト (GB) です。ただし、サービスとレポートによっては、テラバイト (TB) が代わりに表示されることがあります。

使用タイプ

使用タイプ	単位	詳細度	説明
region1-*region2*-AWS-In-A Bytes	GB	毎時	*region2* から *region1* に転送される高速データ量
region1-*region2*-AWS-In-A Bytes-T1	GB	毎時	*region2* から *region1* に転送される T1 高速データの量。T1 は、米国、欧州、および日本の Point of Presence (POPs) への CloudFront リクエストを指します。

使用タイプ	単位	詳細度	説明
region1-*region2*-AWS-In-ABytes-T2	GB	毎時	*region2* から *region1* に転送される T2 高速データの量。T2 は、その他すべての AWS エッジロケーションにある POPs への CloudFront リクエストを指します。
region1-*region2*-AWS-In-Bytes	GB	毎時	*region2* から *region1* に転送されるデータ量
region1-*region2*-AWS-Out-ABytes	GB	毎時	*region2* から *region1* に転送される高速データ量
region1-*region2*-AWS-Out-ABytes-T1	GB	毎時	*region1* から *region2* に転送される T1 高速データの量。T1 は、米国、欧州、および日本の POP への CloudFront リクエストを指します。
region1-*region2*-AWS-Out-ABytes-T2	GB	毎時	*region2* から *region1* に転送される T2 高速データの量。T2 は、その他すべての AWS エッジロケーションにある POP への CloudFront リクエストを指します。
region1-*region2*-AWS-Out-Bytes	GB	毎時	*region2* から *region1* に転送されるデータ量
region-BatchOperations-Jobs	カウント	毎時	実行された S3 バッチオペレーションジョブの数

使用タイプ	単位	詳細度	説明
region-BatchOperations-Objects	カウント	毎時	S3 バッチオペレーションによって実行されたオブジェクトオペレーションの数
region-Bulk-Retrieval-Bytes	GB	毎時	Bulk S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive リクエストで取得したデータの量
region-BytesDeleted-GDA	GB	月別	DeleteObject オペレーションによって S3 Glacier Deep Archive ストレージから削除されたデータの量
region-BytesDeleted-GIR	GB	月別	DeleteObject オペレーションによって S3 Glacier Instant Retrieval ストレージから削除されたデータの量。
region-BytesDeleted-GLACIER	GB	月別	DeleteObject オペレーションによって S3 Glacier Flexible Retrieval ストレージから削除されたデータの量
region-BytesDeleted-INT	GB	月別	DeleteObject オペレーションによって S3 Intelligent-Tiering ストレージから削除されたデータの量

使用タイプ	単位	詳細度	説明
region-BytesDeleted-RRS	GB	月別	DeleteObject オペレーションによって低冗長化ストレージ (RRS) から削除されたデータの量
region-BytesDeleted-SIA	GB	月別	DeleteObject オペレーションによって S3 Standard-IA ストレージから削除されたデータの量
region-BytesDeleted-STANDARD	GB	月別	DeleteObject オペレーションによって S3 Standard ストレージから削除されたデータの量
region-BytesDeleted-ZIA	GB	月別	DeleteObject オペレーションによって S3 One Zone-IA ストレージから削除されたデータの量
region-C3DataTransfer-In-Bytes	GB	毎時	同じ AWS リージョン内で Amazon EC2 から Amazon S3 に転送されたデータの量
region-C3DataTransfer-Out-Bytes	GB	毎時	同じ AWS リージョン内の Amazon S3 から Amazon EC2 に転送されるデータの量
region-CloudFront-In-Bytes	GB	毎時	CloudFront ディストリビューションから AWS リージョン に転送されたデータの量

使用タイプ	単位	詳細度	説明
region-CloudFront-Out-Bytes	GB	毎時	AWS リージョン から CloudFront ディストリビューションに転送されたデータ量
region-DataTransfer-In-Bytes	GB	毎時	インターネットから Amazon S3 に転送されたデータの量
region-DataTransfer-Out-Bytes	GB	毎時	Amazon S3 からインターネット[1] に転送されたデータ量
region-DataTransfer-Regional-Bytes	GB	毎時	同じ AWS リージョン 内で Amazon S3 から AWS リソースに転送されたデータの量
region-EarlyDelete-ByteHrs	GB-時間	毎時	90 日間の最低コミットメントが終了する前に、S3 Glacier Flexible Retrieval ストレージから削除されたオブジェクトの、比例配分されたストレージの使用量 [2]
region-EarlyDelete-GDA	GB-時間	毎時	180 日間の最低コミットメントが終了する前に S3 Glacier Deep Archive ストレージから削除されたオブジェクトの、按分計算されたストレージの使用量2

使用タイプ	単位	詳細度	説明
region-EarlyDelete-GIR	GB-時間	毎時	90 日間の最低コミットメントが終了する前に、S3 Glacier Instant Retrieval から削除されたオブジェクトの比例配分されたストレージの使用量。
region-EarlyDelete-GIR-SmObjects	GB-時間	毎時	90 日間の最低コミットメントが終了する前に S3 Glacier Instant Retrieval から削除された小さなオブジェクト (128 KB未満) の比例配分されたストレージ使用量。
region-EarlyDelete-SIA	GB-時間	毎時	30 日間の最低コミットメントが終了する前に S3 Standard – IA ストレージから削除されたオブジェクトの、按分計算されたストレージの使用状況[3]
region-EarlyDelete-SIA-SmObjects	GB-時間	毎時	30 日間の最低コミットメントが終了する前に S3 Standard – IA ストレージから削除された小さなオブジェクト (128 KB 未満) の、按分計算されたストレージの使用状況[3]

使用タイプ	単位	詳細度	説明
region-EarlyDelete-ZIA	GB-時間	毎時	30 日間の最低コミットメントが終了する前に S3 1 ゾーン – IA ストレージから削除されたオブジェクトの、按分計算されたストレージの使用状況[3]
region-EarlyDelete-ZIA-SmObjects	GB-時間	毎時	30 日間の最低コミットメントが終了する前に S3 1 ゾーン – IA ストレージから削除された小さなオブジェクト (128 KB 未満) の、按分計算されたストレージの使用状況[3]
region-Expedited-Retrieval-Bytes	GB	毎時	Expedited S3 Glacier Flexible Retrieval リクエストを使用して取得されるデータの量
region-Inventory-Objects Listed	オブジェクト	毎時	インベントリリストで、オブジェクトグループごとにリストされるオブジェクトの数 (オブジェクトはバケットまたはプレフィックスごとにグループ化される)
region-Monitoring-Automation-INT	オブジェクト	毎時	S3 Intelligent-Tiering ストレージクラスの監視および自動階層化された一意のオブジェクトの数

使用タイプ	単位	詳細度	説明
region-MRAP-Out-Bytes	GB	毎時	S3 のマルチリージョンアクセスポイントエンドポイントを介してリージョン内のバケットから転送されるデータの量 (MRAP データルーティング料金)。
region-MRAP-In-Bytes	GB	毎時	S3 のマルチリージョンアクセスポイントエンドポイントを介してリージョン内のバケットから転送されるデータの量 (MRAP データルーティング料金)。
regiongroup1-regiongroup2--MRAP-Out-Bytes	GB	毎時	S3 のマルチリージョンアクセスポイントエンドポイントを介して、*regiongroup1* のバケットから AWS ネットワーク外にある *regiongroup2* のクライアントに転送されるデータの量。
regiongroup1-regiongroup2--MRAP-In-Bytes	GB	毎時	S3 のマルチリージョンアクセスポイントエンドポイントを介して、*regiongroup1* のバケットに AWS ネットワーク外にある *regiongroup2* のクライアントから転送されるデータの量。

使用タイプ	単位	詳細度	説明
region-OverwriteBytes-Copy-GDA	GB	月別	CopyObject オペレーションによって S3 Glacier Deep Archive ストレージで上書きされたデータの量
region-OverwriteBytes-Copy-GIR	GB	月別	CopyObject オペレーションによって S3 Glacier Instant Retrieval ストレージで上書きされたデータの量。
region-OverwriteBytes-Copy-GLACIER	GB	月別	CopyObject オペレーションによって S3 Glacier Flexible Retrieval ストレージで上書きされたデータの量
region-OverwriteBytes-Copy-INT	GB	月別	CopyObject オペレーションによって S3 Intelligent-Tiering ストレージで上書きされたデータの量
region-OverwriteBytes-Copy-RRS	GB	月別	CopyObject オペレーションによって低冗長化ストレージ (RRS) で上書きされたデータの量
region-OverwriteBytes-Copy-SIA	GB	月別	CopyObject オペレーションによって S3 Standard-IA ストレージで上書きされたデータの量

使用タイプ	単位	詳細度	説明
region-OverwriteBytes-Copy-STANDARD	GB	月別	CopyObject オペレーションによって S3 Standard ストレージで上書きされたデータの量
region-OverwriteBytes-Copy-ZIA	GB	月別	CopyObject オペレーションによって S3 One Zone-IA ストレージで上書きされたデータの量
region-OverwriteBytes-Put-GDA	GB	月別	PutObject オペレーションによって S3 Glacier Deep Archive ストレージで上書きされたデータの量
region-OverwriteBytes-Put-GIR	GB	月別	PutObject オペレーションによって S3 Glacier Instant Retrieval ストレージで上書きされたデータの量。
region-OverwriteBytes-Put-GLACIER	GB	月別	PutObject オペレーションによって S3 Glacier Flexible Retrieval ストレージで上書きされたデータの量
region-OverwriteBytes-Put-INT	GB	月別	PutObject オペレーションによって S3 Intelligent-Tiering ストレージで上書きされたデータの量

使用タイプ	単位	詳細度	説明
region-OverwriteBytes-Put-RRS	GB	月別	PutObject オペレーションによって低冗長化ストレージ (RRS) で上書きされたデータの量
region-OverwriteBytes-Put-SIA	GB	月別	PutObject オペレーションによって S3 Standard-IA ストレージで上書きされたデータの量
region-OverwriteBytes-Put-STANDARD	GB	月別	PutObject オペレーションによって S3 Standard ストレージで上書きされたデータの量
region-OverwriteBytes-Put-ZIA	GB	月別	PutObject オペレーションによって S3 One Zone-IA ストレージで上書きされたデータの量

使用タイプ	単位	詳細度	説明
region1-*region2*-S3RTC-In-Bytes	GB	月別	PutObjectReplTime 、GetObjectReplTime 、InitiateMultipartUploadReplTime 、UploadPartReplTime 、CompleteMultipartUploadReplTime 、WriteACLReplTime オペレーションによって S3 Replication Time Control (S3 RTC) で *region2* から *region1* に転送されたデータ量

使用タイプ	単位	詳細度	説明
region1-*region2*-S3RTC-Out-Bytes	GB	月別	PutObjectReplTime 、GetObjectReplTime 、InitiateMultipartUploadReplTime 、UploadPartReplTime 、CompleteMultipartUploadReplTime 、WriteACLReplTime オペレーションによって S3 Replication Time Control (S3 RTC) で *region1* から *region2* に転送されたデータ量
region-Requests-GDA-Tier1	カウント	毎時	S3 Glacier Deep Archive オブジェクトの PUT、COPY、POST、CreateMultipartUpload 、UploadPart 、または CompleteMultipartUpload リクエストの数 [6]
region-Requests-GDA-Tier2	カウント	毎時	S3 Glacier Deep Archive オブジェクトの GET および HEAD リクエストの数
region-Requests-GDA-Tier3	カウント	毎時	S3 Glacier Deep Archive 標準復元リクエストの数

使用タイプ	単位	詳細度	説明
region-Requests-GDA-Tier5	カウント	毎時	Bulk S3 Glacier Deep Archive の復元リクエストの数
region-Requests-GIR-Tier1	カウント	毎時	S3 Glacier Instant Retrieval オブジェクトに対する PUT、COPY、または POST リクエストの数。
region-Requests-GIR-Tier2	カウント	毎時	S3 Glacier Instant Retrieval オブジェクトに対する GET および他のすべての S3 Glacier Instant Retrieval-Tier1 以外のリクエストの数。
region-Requests-GLACIER-Tier1	カウント	毎時	S3 Glacier Flexible Retrieval オブジェクトに対する PUT、COPY、POST、CreateMultipartUpload 、UploadPart 、またはCompleteMultipartUpload リクエストの数 [6]
region-Requests-GLACIER-Tier2	カウント	毎時	S3 Glacier Flexible Retrieval オブジェクトに表示されていない GET および他のすべてのリクエストの数

使用タイプ	単位	詳細度	説明
region-Requests-INT-Tier1	カウント	毎時	S3 Intelligent-Tiering オブジェクトに対する PUT、COPY、または POST リクエストの数
region-Requests-INT-Tier2	カウント	毎時	S3 Intelligent-Tiering オブジェクトに対する GET および他のすべての Tier1 以外のリクエストの数
region-Requests-SIA-Tier1	カウント	毎時	S3 Standard–IA オブジェクトに対する PUT、COPY、または POST リクエストの数
region-Requests-SIA-Tier2	カウント	毎時	S3 Standard-IA オブジェクトに対する GET および他のすべての S3 Glacier Instant Retrieval-Tier1 以外のリクエストの数。
region-Requests-Tier1	カウント	毎時	S3 Standard、RRS、タグに対する PUT、COPY、または POST リクエストに加え、すべてのバケットとオブジェクトに対する LIST リクエストの数
region-Requests-Tier2	カウント	毎時	GET および他のすべての Tier1 以外のリクエストの数

使用タイプ	単位	詳細度	説明
region-Requests-Tier3	カウント	毎時	S3 Glacier Flexible Retrieval または S3 Glacier Deep Archiveへのライフサイクルリクエスト、および標準 S3 Glacier Flexible Retrieval 復元リクエストの数。
region-Requests-Tier4	カウント	毎時	S3 Glacier Flexible Retrieval、S3 Intelligent-Tiering、S3 Standard-IA、または S3 One Zone-IA ストレージへのライフサイクル移行数
region-Requests-Tier5	カウント	毎時	Bulk S3 Glacier Flexible Retrieval 復元リクエストの数
region-Requests-Tier6	カウント	毎時	Expedited S3 Glacier Flexible Retrieval 復元リクエストの数
region-Requests-Tier8	カウント	毎時	S3 Access Grants リクエストの数
region-Requests-XZ-Tier1	カウント	毎時	S3 Express One Zone オブジェクトに対する PUT または COPY リクエストの数
region-Requests-XZ-Tier2	カウント	毎時	S3 Express One Zone オブジェクトに対する GET およびその他すべての S3 Express One Zone-Tier1 以外のリクエストの数

使用タイプ	単位	詳細度	説明
region-Requests-ZIA-Tier1	カウント	毎時	S3 One Zone-IA オブジェクトに対する PUT、COPY、または POST リクエストの数
region-Requests-ZIA-Tier2	カウント	毎時	S3 One Zone-IA オブジェクトに対する GET および他のすべての S3 One Zone-IA-Tier1 以外のリクエストの数
region-Retrieval-GIR	GB	毎時	S3 Glacier Instant Retrieval ストレージから取得したデータの量。
region-Retrieval-SIA	GB	毎時	S3 Standard – IA ストレージから取得したデータの量
region-Retrieval-XZ	GB	毎時	S3 Express One Zone ストレージでの特定の取り出しリクエスト (PUT または COPY) で 512 KB を超えるデータの一部
region-Retrieval-ZIA	GB	毎時	S3 1 ゾーン – IA ストレージから取得したデータの量
region-S3DSSE-In-Bytes	GB	月別	Amazon S3 によって二重暗号化されたデータの量
region-S3DSSE-Out-Bytes	GB	月別	Amazon S3 によって復号された二重暗号化されたデータの量

使用タイプ	単位	詳細度	説明
region-S3G-DataTransfer-In-Bytes	GB	毎時	S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージからオブジェクトを復元するために Amazon S3 に転送されるデータの量
region-S3G-DataTransfer-Out-Bytes	GB	毎時	オブジェクトを S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージに移行するために Amazon S3 から転送されたデータの量
region-Select-Returned-Bytes	GB	毎時	S3 Standard ストレージから Select リクエストを使用して返されたデータの量
region-Select-Returned-GIR-Bytes	GB	毎時	S3 Glacier Instant Retrieval ストレージから選択リクエストを使用して返されたデータの量
region-Select-Returned-INT-Bytes	GB	毎時	S3 Intelligent-Tiering ストレージから Select リクエストを使用して返されたデータの量
region-Select-Returned-SIA-Bytes	GB	毎時	S3 Standard – IA ストレージから Select リクエストを使用して返されたデータの量

使用タイプ	単位	詳細度	説明
region-Select-Returned-ZIA-Bytes	GB	毎時	S3 1 ゾーン – IA ストレージから Select リクエストを使用して返されたデータの量
region-Select-Scanned-Bytes	GB	毎時	S3 Standard ストレージから Select リクエストを使用してスキャンされたデータの量
region-Select-Scanned-GIR-Bytes	GB	毎時	S3 Glacier Instant Retrieval ストレージから選択リクエストを使用してスキャンされたデータの量。
region-Select-Scanned-INT-Bytes	GB	毎時	S3 Intelligent-Tiering ストレージから Select リクエストを使用してスキャンされたデータの量
region-Select-Scanned-SIA-Bytes	GB	毎時	S3 Standard – IA ストレージから Select リクエストを使用してスキャンされたデータの量
region-Select-Scanned-ZIA-Bytes	GB	毎時	S3 1 ゾーン – IA ストレージから Select リクエストを使用してスキャンされたデータの量
region-Standard-Retrieval-Bytes	GB	毎時	標準 S3 Glacier Instant Retrieval または S3 Glacier Deep Archive リクエストで取得したデータの量

使用タイプ	単位	詳細度	説明
region-StorageAnalytics-ObjCount	オブジェクト	毎時	各ストレージクラス分析設定でモニタリングされる一意のオブジェクトの数。
region-StorageLens-ObjCount	オブジェクト	1 日 1 回	S3 Storage Lens の高度なメトリクスとレコメンデーションによって追跡される、各 S3 Storage Lens ダッシュボードの一意のオブジェクトの数。
region-StorageLensFreeTier-ObjCount	オブジェクト	1 日 1 回	S3 Storage Lens の使用状況メトリクスによって追跡される、各 S3 Storage Lens ダッシュボードの一意のオブジェクトの数。
StorageObjectCount	カウント	1 日 1 回	特定のバケット内に保存されたオブジェクトの数
region-TagStorage-TagHrs	タグ時間	1 日 1 回	時間毎にレポートされる、バケット内のすべてのオブジェクトのタグの合計
region-TimedStorage-ByteHrs	GB-月	1 日 1 回	S3 Standard ストレージに保存されたデータの GB-月の数
region-TimedStorage-GDA-ByteHrs	GB-月	1 日 1 回	S3 Glacier Deep Archive ストレージに保存されたデータの GB-月の数

使用タイプ	単位	詳細度	説明
region-TimedStorage-GDA-Staging	GB-月	1 日 1 回	S3 Glacier Deep Archive ステージングストレージに保存されたデータの GB-月の数
region-TimedStorage-GIR-ByteHrs	GB-月	1 日 1 回	S3 Glacier Instant Retrieval ストレージに保存されたデータの GB-月の数。
region-TimedStorage-GIR-SmObjects	GB-月	1 日 1 回	S3 Glacier Flexible Retrieval の IA ストレージに保存された小さなオブジェクト (128 KB 未満) の GB-月の数
region-TimedStorage-GlacierByteHrs	GB-月	1 日 1 回	S3 Glacier Flexible Retrieval ストレージに保存されたデータの GB-月の数
region-TimedStorage-GlacierStaging	GB-月	1 日 1 回	S3 Glacier Flexible Retrieval staging ストレージに保存されたデータの GB-月の数
region-TimedStorage-INT-FA-ByteHrs	GB-月	1 日 1 回	S3 Intelligent-Tiering ストレージの高頻度のアクセス階層に保存されたデータの GB-月の数[5]
region-TimedStorage-INT-IA-ByteHrs	GB-月	1 日 1 回	S3 Intelligent-Tiering ストレージの小頻度のアクセス階層に保存されたデータの GB-月の数

使用タイプ	単位	詳細度	説明
region-TimedStorage-INT-AA-ByteHrs	GB-月	1 日 1 回	S3 Intelligent-Tiering ストレージのアーカイブのアクセス階層に保存されたデータの GB-月の数
region-TimedStorage-INT-AIA-ByteHrs	GB-月	1 日 1 回	S3 Intelligent-Tiering ストレージのアーカイブのインスタントアクセス階層に保存されたデータの GB-月の数
region-TimedStorage-INT-DAA-ByteHrs	GB-月	1 日 1 回	S3 Intelligent-Tiering ストレージの Deep アーカイブのアクセス階層に保存されたデータの GB-月の数
region-TimedStorage-RRS-ByteHrs	GB-月	1 日 1 回	低冗長化ストレージ (RRS) に保存されたデータの GB-月の数
region-TimedStorage-SIA-ByteHrs	GB-月	1 日 1 回	S3 Standard-IA ストレージに保存されたデータの GB-月の数
region-TimedStorage-SIA-SmObjects	GB-月	1 日 1 回	S3 Standard – IA ストレージに保存された小さなオブジェクト (128 KB 未満) の GB-月の数[4]
region-TimedStorage-XZ-ByteHrs	GB-月	1 日 1 回	S3 Express One Zone ストレージに保存されたデータの GB-月の数

使用タイプ	単位	詳細度	説明
region-TimedStorage-ZIA-ByteHrs	GB-月	1 日 1 回	S3 1 ゾーン – IA ストレージに保存されたデータの GB-月の数
region-TimedStorage-ZIA-SmObjects	GB-月	1 日 1 回	S3 One Zone-IA ストレージに保存された小さなオブジェクト (128 KB 未満) の GB-月の数
region-Upload-XZ	GB	毎時	S3 Express One Zone での特定のアップロードリクエスト (PUT または COPY) で 512 KB を超えるデータの量

メモ

1. 完了前に転送を終了すると、転送されるデータ量がアプリケーションが受け取るデータ量を超える可能性があります。この不一致は、転送終了リクエストを即時に実行することができず、終了リクエストの実行が保留されている間にある程度の量のデータが転送中である可能性があるために発生する可能性があります。この転送中のデータは、「アウト」された転送データとして請求されます。
2. S3 Glacier または S3 Glacier Deep Archive ストレージクラスにアーカイブされたオブジェクトは、最小ストレージコミットメントが経過する前に削除または上書きされるか、別のストレージクラスに移行されます (S3 Glacier Instant Retrieval および S3 Glacier Flexible Retrieval の場合は 90 日、S3 Glacier Deep Archive の場合は 180 日)。残りの日数に対しては、ギガバイト単位で比例配分された料金が適用されます。
3. S3 Standard-IA または S3 One Zone-IA ストレージのオブジェクトを 30 日より前に削除、上書き、または他のストレージクラスに移行すると、残りの日数に対してはギガバイト単位で比例配分された料金が適用されます。
4. S3 Standard-IA または S3 One Zone-IA ストレージの小さい (128 KB より小さい) オブジェクトを 30 日より前に削除、上書き、または他のストレージクラスに移行すると、残りの日数に対してはギガバイト単位で按分計算された料金が適用されます。

5. S3 Intelligent-Tiering ストレージクラスのオブジェクトには、請求可能な最小オブジェクトサイズはありません。128 KB 未満のオブジェクトは、モニタリングされないか、自動階層化の対象となります。小さいオブジェクトは必ず S3 Intelligent-Tiering 高頻度アクセス階層に保存されます。
6. S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスのいずれかに対して「CreateMultipartUpload」、「UploadPart」、または「UploadPartCopy」リクエストを開始すると、マルチパートアップロードが完了するまで、リクエストは S3 Standard のリクエスト料金で請求されます。アップロードが完了すると、送信先の S3 Glacier ストレージの PUT 料金で 1 つの CompleteMultipartUpload リクエストが請求されます。S3 Glacier Flexible Retrieval ストレージクラスへの PUT の処理中のマルチパートアップロードパートは、アップロードが完了するまで S3 Glacier Flexible Retrieval ステージングストレージとして S3 Standard ストレージ料金で請求されます。同様に、S3 Glacier Deep Archive ストレージクラスへの PUT のための処理中のマルチパートアップロードパートは、アップロードが完了するまで S3 Glacier Deep Archive ステージングストレージとして S3 Standard ストレージ料金で請求されます。
7. S3 Express One Zone では、リクエストサイズが 512 KB 以下の場合、リクエストごとの定額料金が適用されます。リクエストの 512 KB を超える部分の PUT リクエストと GET リクエストには、GB あたりの追加料金が適用されます。
8. S3 Express One Zone ストレージクラスでサポートされている機能については、「S3 Express One Zone がサポートしていない Amazon S3 の機能」を参照してください。
9. GB 単位で請求される単位の使用タイプは、使用状況レポートでバイト単位で計算されます。
10. GB-月は、GB-時間の数を 1 か月で集計し、その月の時間数で割って算出されます。詳細については、「Amazon S3 のよくある質問: Amazon S3 の利用には、どのように課金され、請求されますか?」を参照してください。

> **Note**
>
> 一般に、S3 バケット所有者は、HTTP 200 OK 成功レスポンスと HTTP 4XX クライアントエラーレスポンスを含むリクエストに対して課金されます。バケット所有者は、HTTP 503 Slow Down エラーなどの HTTP 5XX サーバーエラーレスポンスに対しては課金されません。HTTP 3XX の S3 エラーコードと請求されない 4XX ステータスコードの詳細については、「Amazon S3 のエラーレスポンスに対する請求」を参照してください。バケットがリクエスタ支払いバケットとして設定されている場合の請求の詳細については、「リクエスタ支払いの課金の仕組み」を参照してください。

使用状況レポートの追跡オペレーション

オペレーションは、お役様の AWS オブジェクトまたはバケットで実行されるアクションを、特定の使用タイプによって記述します。オペレーションは、PutObject や ListBucket のような、見ればすぐわかるコードによって示されます。特定の使用タイプがバケットのどのアクションで発生しているかを確認するには、これらのコードを使用します。使用状況レポートを作成する場合、レポートに [All Operations] (すべてのオペレーション) を含めるか、GetObject など特定のオペレーションのみを含めるかを選択できます。

詳細情報

- AWSAmazon S3 の 使用状況レポート
- Amazon S3 の AWS Billing レポート
- Amazon S3 の料金
- Amazon S3 のよくある質問

Amazon S3 のエラーレスポンスに対する請求

> **Important**
>
> 2024 年 5 月 13 日に、バケット所有者によって開始されていない不正なリクエストに対する料金を排除する変更のデプロイを開始しました。この変更のデプロイが完了すると、バケット所有者は、個々の AWS アカウントまたは AWS 組織の外部からリクエストが開始されたときに AccessDenied (HTTP 403 Forbidden) エラーを返すリクエストに対してリクエストまたは帯域幅料金を請求されることはありません。現在のページには、請求されない HTTP 3XX および 4XX ステータスコードの全一覧が表示されています。この請求の変更は、アプリケーションの更新を必要とせず、すべての S3 バケットに適用されます。すべての AWS リージョンでこの変更のデプロイが完了したら、ドキュメントを更新します。

一般に、S3 バケット所有者は、HTTP 200 OK 成功レスポンスと HTTP 4XX クライアントエラーレスポンスを含むリクエストに対して課金されます。バケット所有者は、HTTP 503 Slow Down エラーなどの HTTP 5XX サーバーエラーレスポンスに対しては課金されません。バケットがリクエスタ支払いバケットとして設定されている場合の請求の詳細については、「リクエスタ支払いの課金の仕組み」を参照してください。

次の表は、HTTP 3XX および 4XX ステータスコードで、請求されない特定のエラーコードの一覧です。ウェブサイトホスティングで設定されたバケットでは、S3 がカスタムエラードキュメントを返した場合、またはカスタムリダイレクトを行った場合は、該当するリクエストおよびその他の料金が適用されます。

> **Note**
>
> `AccessDenied` (HTTP `403 Forbidden`) の場合、リクエストがバケット所有者の個々の AWS アカウントまたはバケット所有者の AWS 組織外で開始されても、S3 ではバケット所有者への請求を行いません。

HTTP ステータスコード	エラーコード	エラーコードの説明	
301 Moved Permanently	PermanentRedirect	アクセスしようとしているバケットは、指定されたエンドポイントを使用してアドレス指定する必要があります。今後のリクエストはこのエンドポイントに送信してください。	
	PermanentRedirectControlError	アクセスしようとしている API 操作は、指定されたエンドポイントを使用してアドレス指定する必要があります。今後のリクエストはこのエンドポイントに送信してください。	
一時的なダイレクト (302)	TemporaryRedirect	ドメインネームシステム (DNS) サーバーが更新されている間、このバケッ	

HTTP ステータスコード	エラーコード	エラーコードの説明
		トにリダイレクトされています。
400 Bad Request	AuthorizationHeaderMalformed	指定した認証ヘッダーが無効です。
	AuthorizationQueryParametersError	指定した認証クエリパラメータが無効です。
	ExpiredToken	提供されたトークンの有効期限が切れています。
	IllegalLocationConstraintException	バケットが存在するリージョンとは異なるリージョンからバケットにアクセスしようとしています。このエラーを回避するには、`--region` オプションを使用します。例: `aws s3 cp awsexample.txt s3://example-s3-bucket/ --region ap-east-1` 。

HTTP ステータスコード	エラーコード	エラーコードの説明	
	InvalidArgument	このエラーは、次の原因によって発生する可能性があります。 • 指定された引数は有効ではありませんでした。 • リクエストに必要なヘッダーがありませんでした。 • 指定された引数が不完全であるか、形式が正しくありませんでした。 • 指定された引数は、3 以上の長さである必要があります。	
	InvalidDigest	指定した Content-MD5 またはチェックサム値が無効です。	
	InvalidEncryptionAlgorithmError	指定した暗号化リクエストが無効です。有効値は AES256 です。	

HTTP ステータスコード	エラーコード	エラーコードの説明	
	InvalidRequest	このエラーは、次の原因によって発生する可能性があります。 • リクエストで間違った署名バージョンを使用しています。AWS4-HMAC-SHA256 (署名バージョン 4) を使用してください。 • アクセスポイントは、既存のバケットに対してのみ作成できます。 • アクセスポイントが削除可能な状態ではありません。 • アクセスポイントは、既存のバケットに対してのみ作成できます。 • 次のトークンが無効です。 • ライフサイクルルールで少なくとも 1 つのアクションを指定する必要があります。 • 少なくとも 1 つのライフサイクルルールを	

HTTP ステータスコード	エラーコード	エラーコードの説明	
		指定する必要があります。 • ライフサイクルルールの数は、許可されている 1000 ルールの制限を超えてはなりません。 • `MaxResults` パラメータの範囲が無効です。 • SOAP リクエストは HTTPS 接続を介して行う必要があります。 • Amazon S3 Transfer Acceleration は、DNS に準拠していない名前のバケットではサポートされていません。 • Amazon S3 Transfer Acceleration は、ピリオド (.) が名前に含まれているバケットではサポートされていません。 • Amazon S3 Transfer Acceleration エンドポイントは、仮想スタイ	

HTTP ステータスコード	エラーコード	エラーコードの説明	
		ルのリクエストのみをサポートします。 • Amazon S3 Transfer Acceleration は、このバケットでは設定されていません。 • Amazon S3 Transfer Acceleration は、このバケットでは無効になっています。 • Amazon S3 Transfer Acceleration は、このバケットではサポートされていません。サポートが必要な場合は、AWS Support にお問い合わせください。 • Amazon S3 Transfer Acceleration はこのバケットでは有効にできません。サポートが必要な場合は、AWS Support にお問い合わせください。 • HTTP ヘッダーとクエリパラメータで指定された値が競合しています。	

HTTP ステータスコード	エラーコード	エラーコードの説明	
		• HTTP ヘッダーと POST フォームフィールドで指定された値が競合しています。 • サイズが 5GB を超えるオブジェクトに対して CopyObject 要求が行われました。	
	InvalidSOAPRequest	SOAP リクエスト本文が無効です。	
	InvalidStorageClass	指定されたストレージクラスが無効です。	
	InvalidTag	リクエストに無効なタグ入力が含まれています。例えば、リクエストに重複するキー、長すぎるキーまたは値、またはシステムタグが含まれている可能性があります。	
	InvalidToken	指定されたトークンは形式に誤りがあるか無効です。	
	InvalidURI	指定された URI を解析できませんでした。	
	KeyTooLongError	キーが長すぎます。	

HTTP ステータスコード	エラーコード	エラーコードの説明	
	MalformedACLError	指定された ACL の書式に誤りがあるか、AWS の公開済みスキーマに一致しませんでした。	
	MalformedPOSTRequest	POST リクエストの本文が正しい形式の multipart/form-data ではありません。	
	MalformedXML	指定された XML の書式に誤りがあるか、Amazon の公開済みスキーマに一致しませんでした。	
	MaxPostPreDataLengthExceededError	アップロードファイルの前の POST リクエストフィールドが大きすぎます。	
	MetadataTooLarge	メタデータヘッダーが許容されるメタデータの最大サイズを超えています。	
	MissingRequestBodyError	空の XML ドキュメントをリクエストとして送信しました。	
	MissingSecurityHeader	リクエストに必要なヘッダーがありません。	

HTTP ステータスコード	エラーコード	エラーコードの説明	
	NoLoggingStatusForKey	キーのログ記録ステータスサブリソースのようなものはありません。	
	RequestHeaderSectionTooLarge	The request header and query parameters used to make the request exceed the maximum allowed sizes	
	UnexpectedContent	This request contains unsupported content.	
	UserKeyMustBeSpecified	バケット POST リクエストは、指定されたフィールド名を含んでいなければなりません。指定されている場合は、フィールドの順序を確認してください。	
	IncorrectEndpoint	指定されたバケットが別のリージョンに存在します。リクエストを正しいエンドポイントに直接送信してください。	
403 Forbidden	RequestTimeTooSkewed	リクエスト時間とサーバー時間の差が大きすぎます。	

HTTP ステータスコード	エラーコード	エラーコードの説明	
	SignatureDoesNotMatch	計算したリクエスト署名が、指定された署名と一致しません。AWS シークレットアクセスキーと署名方法を確認してください。詳細については、「REST リクエストの署名と認証」および「SOAP リクエストの認証方法」を参照してください。	
	NotSignedUp	アカウントが Amazon S3 に対してサインアップされていません。Amazon S3 を使用する前に、サインアップする必要があります。サインアップは https://aws.amazon.com/s3 で行うことができます。	
	InvalidSecurity	提供されたセキュリティ認証情報が無効です。	
	InvalidPayer	このオブジェクトへのすべてのアクセスが無効になっています。詳細については、「AWS に問い合わせる」を参照してください。	

HTTP ステータスコード	エラーコード	エラーコードの説明
	InvalidAccessKeyId	指定された AWS アクセスキー ID が見つかりません。
	AccountProblem	AWS アカウント アカウントに問題があり、それによりオペレーションを正常に完了することができません。詳細については、「AWS に問い合わせる」を参照してください。
	UnauthorizedAccessError	中国リージョンにのみ適用されます。ICP ライセンスを持たないバケットに対してリクエストが行われたときに返されます。詳細については、「ICP Recordal」を参照してください。
404 Not Found	NoSuchUpload	指定されたマルチパートアップロードは存在しません。アップロード ID が無効であるか、マルチパートアップロードが中止または完了した可能性があります。
	NoSuchWebsiteConfiguration	指定されたバケットにはウェブサイト設定がありません。

HTTP ステータスコード	エラーコード	エラーコードの説明
405 Method Not Allowed	MethodNotAllowed	指定されたメソッドは、このリソースに対して許可されていません。
409 Conflict	BucketAlreadyExists	リクエストされたバケット名は使用できません。バケット名前空間は、システムのすべてのユーザーによって共有されています。別の名前を使用してもう一度試してください。
	InvalidBucketState	リクエストはバケットの現在の状態で有効ではありません。
	OperationAborted	競合する条件付きオペレーションが現在このリソースに対して進行中です。もう一度試してください。
411 Length Required	MissingContentLength	Content-Length HTTP ヘッダーを指定する必要があります。
412 Precondition Failed	RequestIsNotMultiPartContent	バケット POST リクエストは、enctype が multipart/form-data である必要があります。

Amazon S3 Select を使用したデータのフィルタリングと取得

Amazon S3 Select では、構造化クエリ言語 (SQL) ステートメントを使用して Amazon S3 オブジェクトのコンテンツをフィルタリングし、必要なデータのサブセットのみ取得できます。Amazon S3 Select を使用してこのデータをフィルタリングする場合、Amazon S3 が転送するデータの量を抑えることで、このデータの取得に必要なコストを削減し、レイテンシーを抑えることができます。

Amazon S3 Select で一度にクエリできるのは、1 つのオブジェクトのみです。CSV、JSON、または Apache Parquet 形式で保存されたオブジェクトのみが対象となります。また、GZIP や BZIP2 で圧縮されたオブジェクト (CSV と JSON 形式のオブジェクトのみ) や、サーバー側で暗号化されたオブジェクトにも使用できます。結果の形式 (CSV または JSON) や、結果のレコードを区切る方法は指定できます。

リクエストで SQL 式を Amazon S3 に渡します。Amazon S3 Select は SQL のサブセットをサポートします。Amazon S3 Select でサポートされている SQL 要素の詳細については、「Amazon S3 Select の SQL リファレンス」を参照してください。

SQL クエリは、Amazon S3 コンソール、AWS Command Line Interface、(AWS CLI)、`SelectObjectContent` REST API オペレーション、または AWS SDK を使用して実行できます。

> **Note**
>
> Amazon S3 コンソールでは、返されるデータの量が 40 MB に制限されます。より多くのデータを取得するには、AWS CLI または API を使用します。

要件と制限

Amazon S3 Select を使用するための要件は以下のとおりです。

- クエリ対象のオブジェクトに対して `s3:GetObject` アクセス許可が必要です。
- クエリ対象のオブジェクトが、お客様が指定したキーによるサーバー側の暗号化 (SSE-C) で暗号化されている場合は、`https` を使用する必要があり、さらにリクエストで暗号化キーを提供する必要があります。

Amazon S3 Select を使用するときは以下の制限が適用されます。

- S3 Select では、リクエストごとにクエリできるのは 1 つのオブジェクトのみです。
- SQL 式の最大長は 256 KB です。
- 入力または結果のレコードの最大長は 1 MB です。
- Amazon S3 Select は JSON 出力形式を使用してネストされたデータのみを出力できます。
- S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive、または Reduced Redundancy Storage (RRS) のストレージクラスに保存されたオブジェクトのクエリはできません。また、S3 Intelligent-Tiering Archive アクセス階層や S3 Intelligent-Tiering Deep Archive アクセス階層に保存されたオブジェクトもクエリできません。ストレージクラスの詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。

Amazon S3 Select を Parquet オブジェクトで使用する場合は、次のとおりの追加の制限が適用されます。

- Amazon S3 Select は、GZIP または Snappy を使用した列指向の圧縮のみをサポートしています。Amazon S3 Select は、Parquet オブジェクトのオブジェクト全体の圧縮はサポートしていません。
- Amazon S3 Select は Parquet 出力をサポートしていません。出力形式として CSV または JSON を指定する必要があります。
- 最大非圧縮の行グループサイズは 512 MB です。
- オブジェクトのスキーマに指定されているデータ型を使用する必要があります。
- 繰り返しフィールドを選択すると、最後の値のみが返されます。

リクエストの構築

リクエストを構築するときは、`InputSerialization` オブジェクトを使用してクエリ対象のオブジェクトの詳細を指定します。`OutputSerialization` オブジェクトを使用して結果を返す方法の詳細を指定します。Amazon S3 でリクエストをフィルタリングできるように、SQL 式を含めることもできます。

Amazon S3 Select リクエストの構築の詳細については、「Amazon Simple Storage Service API リファレンス」の「SelectObjectContent」を参照してください。また、以下のセクションでいずれかの SDK コードサンプルを参照できます。

スキャン範囲を使用したリクエスト

Amazon S3 Select を利用すると、クエリするバイト範囲を指定して、オブジェクトのサブセットをスキャンできます。この機能により、処理を一連の重複していないスキャン範囲の個別の Amazon S3 Select リクエストに分割して、オブジェクト全体のスキャンを並列化できます。

スキャン範囲をレコードの境界に合わせる必要はありません。Amazon S3 Select スキャン範囲リクエストは、指定したバイト範囲全体で実行されます。レコードが指定したスキャン範囲内で開始しても、そのスキャン範囲を超える場合は、クエリによって処理されます。例えば、次に示す Amazon S3 オブジェクトには、一連のレコードが行区切りの CSV 形式で含まれています。

```
A,B
C,D
D,E
E,F
G,H
I,J
```

Amazon S3 Select の `ScanRange` パラメータを使用して、スキャンを 1 (バイト) で開始し、4 (バイト) で終了するとします。この場合、スキャン範囲は「`,`」で開始し、`C` で始まるレコードの最後までスキャンします。スキャン範囲のリクエストは、結果として、レコードの最後までの `C, D` を返します。

Amazon S3 Select のスキャン範囲リクエストは、Parquet、CSV (引用符区切りなし)、JSON オブジェクト (`LINES` モードのみ) をサポートしています。CSV および JSON オブジェクトは非圧縮である必要があります。行ベースの CSV および JSON オブジェクトの場合、スキャン範囲が Amazon S3 Select リクエストの一部として指定されると、スキャン範囲内で開始するすべてのレコードが処理されます。Parquet オブジェクトの場合、リクエストされたスキャン範囲内で開始するすべての行グループが処理されます。

Amazon S3 Select のスキャン範囲リクエストは、AWS CLI、Amazon S3 API、AWS SDK で使用できます。Amazon S3 Select リクエストで、この機能について、`ScanRange` パラメータを使用できます。詳細については、「Amazon Simple Storage Service API リファレンス」の「SelectObjectContent」を参照してください。

エラー

クエリの実行中に問題が発生すると、Amazon S3 Select からエラーコードおよび関連するエラーメッセージが返されます。エラーコードの一覧とその説明については、Amazon Simple Storage

Service API リファレンスのエラーコードページの「SELECT Object Content エラーコードのリスト」セクションを参照してください。

Amazon S3 Select の詳細については、以下のトピックを参照してください。

トピック

- オブジェクトでの Amazon S3 Select の使用例
- Amazon S3 Select の SQL リファレンス

オブジェクトでの Amazon S3 Select の使用例

S3 Select を使用すると、Amazon S3 コンソール、REST API、AWS SDK を使用して単一のオブジェクトのコンテンツを選択できます。

S3 でサポートされている SQL 関数の詳細については、「SQL 関数」を参照してください。

S3 コンソールの使用

Amazon S3 コンソールでオブジェクトからコンテンツを選択するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. コンテンツを選択するオブジェクトが含まれているバケットを選択し、次にオブジェクトの名前を選択します。
4. [オブジェクトアクション] を選択し、[S3 Select を使用したクエリ] を選択します。
5. 入力データの形式に基づいて [入力設定] を行います。
6. 受信する出力の形式に基づいて、[出力設定] を行います。
7. 選択したオブジェクトからレコードを抽出するには、[SQL クエリ] に SELECT SQL コマンドを入力します。SQL コマンドの記述方法の詳細については、「Amazon S3 Select の SQL リファレンス」を参照してください。
8. SQL クエリを入力したら、[SQL クエリを実行] を選択します。次に、[クエリ結果] で SQL クエリの結果を確認できます。

REST API の使用

AWS SDK を使用して、オブジェクトのコンテンツを選択できます。ただし、アプリケーションで必要な場合は、REST リクエストを直接送信できます。リクエストとレスポンスの形式の詳細については、「SelectObjectContent」を参照してください。

AWS SDK の使用

Amazon S3 Select で `selectObjectContent` メソッドを使用してオブジェクトのコンテンツの一部を選択できます。このメソッドが成功すると、SQL 式の結果が返されます。

Java

次の Java コードでは、CSV 形式で保存されているデータを含むオブジェクトに保存されている各レコードの最初の列の値が返ります。また、`Progress` メッセージおよび `Stats` メッセージが返るようリクエストされます。CSV 形式のデータを含む有効なバケット名とオブジェクトを指定する必要があります。

作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
package com.amazonaws;

import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CSVInput;
import com.amazonaws.services.s3.model.CSVOutput;
import com.amazonaws.services.s3.model.CompressionType;
import com.amazonaws.services.s3.model.ExpressionType;
import com.amazonaws.services.s3.model.InputSerialization;
import com.amazonaws.services.s3.model.OutputSerialization;
import com.amazonaws.services.s3.model.SelectObjectContentEvent;
import com.amazonaws.services.s3.model.SelectObjectContentEventVisitor;
import com.amazonaws.services.s3.model.SelectObjectContentRequest;
import com.amazonaws.services.s3.model.SelectObjectContentResult;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.concurrent.atomic.AtomicBoolean;

import static com.amazonaws.util.IOUtils.copy;
```

```
/**
 * This example shows how to query data from S3Select and consume the response in
 the form of an
 * InputStream of records and write it to a file.
 */

public class RecordInputStreamExample {

    private static final String BUCKET_NAME = "${my-s3-bucket}";
    private static final String CSV_OBJECT_KEY = "${my-csv-object-key}";
    private static final String S3_SELECT_RESULTS_PATH = "${my-s3-select-results-
path}";
    private static final String QUERY = "select s._1 from S3Object s";

    public static void main(String[] args) throws Exception {
        final AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();

        SelectObjectContentRequest request = generateBaseCSVRequest(BUCKET_NAME,
 CSV_OBJECT_KEY, QUERY);
        final AtomicBoolean isResultComplete = new AtomicBoolean(false);

        try (OutputStream fileOutputStream = new FileOutputStream(new File
 (S3_SELECT_RESULTS_PATH));
             SelectObjectContentResult result =
 s3Client.selectObjectContent(request)) {
            InputStream resultInputStream =
 result.getPayload().getRecordsInputStream(
                    new SelectObjectContentEventVisitor() {
                        @Override
                        public void visit(SelectObjectContentEvent.StatsEvent event)
                        {
                            System.out.println(
                                    "Received Stats, Bytes Scanned: " +
 event.getDetails().getBytesScanned()
                                            +  " Bytes Processed: " +
 event.getDetails().getBytesProcessed());
                        }

                        /*
                         * An End Event informs that the request has finished
 successfully.
                         */
                        @Override
```

```
                        public void visit(SelectObjectContentEvent.EndEvent event)
                        {
                            isResultComplete.set(true);
                            System.out.println("Received End Event. Result is
 complete.");
                        }
                    }
            );

            copy(resultInputStream, fileOutputStream);
        }

        /*
         * The End Event indicates all matching records have been transmitted.
         * If the End Event is not received, the results may be incomplete.
         */
        if (!isResultComplete.get()) {
            throw new Exception("S3 Select request was incomplete as End Event was
 not received.");
        }
    }

    private static SelectObjectContentRequest generateBaseCSVRequest(String bucket,
 String key, String query) {
        SelectObjectContentRequest request = new SelectObjectContentRequest();
        request.setBucketName(bucket);
        request.setKey(key);
        request.setExpression(query);
        request.setExpressionType(ExpressionType.SQL);

        InputSerialization inputSerialization = new InputSerialization();
        inputSerialization.setCsv(new CSVInput());
        inputSerialization.setCompressionType(CompressionType.NONE);
        request.setInputSerialization(inputSerialization);

        OutputSerialization outputSerialization = new OutputSerialization();
        outputSerialization.setCsv(new CSVOutput());
        request.setOutputSerialization(outputSerialization);

        return request;
    }
}
```

JavaScript

AWS SDK for JavaScript と S3 `SelectObjectContent` API オペレーションを使用して Amazon S3 に保存された JSON ファイルと CSV ファイルからレコードを選択する JavaScript の例については、ブログ記事「AWS SDK for JavaScript における Amazon S3 Select のサポートの紹介」を参照してください。

Python

S3 Select を使用してカンマ区切り値 (CSV) ファイルとして Amazon S3 にロードされたデータを検索する SQL クエリの使用に関する Python の例については、ブログ記事「Amazon S3 Select を使用したサーバーまたはデータベースなしのデータのクエリ」を参照してください。

Amazon S3 Select の SQL リファレンス

このリファレンスには、Amazon S3 Select でサポートされる構造化クエリ言語 (SQL) の説明が記載されています。

トピック

SELECT コマンド

Amazon S3 セレクトは、`SELECT` SQL コマンドのみをサポートします。次の ANSI スタンダードの句は、`SELECT` 用にサポートされています。

- `SELECT` リスト
- `FROM` 句
- `WHERE` 句
- `LIMIT` 句

> **Note**
>
> Amazon S3 Select クエリは、現在サブクエリや結合はサポートしていません。

SELECT のリスト

SELECT リストは、クエリに返させる列、関数、および式を指定します。このリストは、クエリの出力を表しています。

```
SELECT *
SELECT projection1 AS column_alias_1, projection2 AS column_alias_2
```

SELECT (アスタリスク) を使用した最初の SELECT 形式では、WHERE 句を通過したすべての行がそのまま返されます。2 番目の SELECT 形式では、列ごとにユーザー定義の出力スカラー式 ***projection1*** および ***projection2*** を含む行が作成されます。

FROM 句

Amazon S3 Select では、FROM 句の以下の形式がサポートされています。

```
FROM table_name
FROM table_name alias
FROM table_name AS alias
```

FROM 句の各形式で、table_name はクエリの対象となる S3Object です。従来のリレーショナルデータベースから移行したユーザーは、これをテーブルに対する複数のビューを含むデータベーススキーマとして考えることができます。

スタンダード SQL に従って、FROM 句は WHERE 句でフィルタリングされ SELECT リストで射影される行を作成します。

Amazon S3 Select に格納されている JSON オブジェクトでは、次の形式の FROM 句も使用することができます。

```
FROM S3Object[*].path
FROM S3Object[*].path alias
FROM S3Object[*].path AS alias
```

この形式の FROM 句を使用すると、JSON オブジェクト内の配列またはオブジェクトから選択できます。以下のいずれかの形式で path を指定できます。

- 名前 (オブジェクト内): `.name` または `['name']`
- インデックス (配列内): `[index]`
- ワイルドカード文字 (オブジェクト内): `.*`
- ワイルドカード文字 (配列内): `[*]`

> **Note**
>
> - この形式の FROM 句は JSON オブジェクトに対してのみ機能します。
> - ワイルドカード文字は常に最低 1 つのレコードを発行します。レコードが一致しない場合、Amazon S3 Select は値 MISSING を発行します。出力のシリアル化中 (クエリの実行の完了後)、Amazon S3 Select は MISSING 値を空のレコードに置き換えます。
> - 集約関数 (AVG、COUNT、MAX、MIN、および SUM) は、MISSING 値をスキップします。
> - エイリアスを指定せずにワイルドカード文字を使用する場合は、パスの最後の要素を使用して行を参照できます。例えば、クエリ `SELECT price FROM S3Object[*].books[*].price` を使用して、ブックのリストからすべての価格を選択できます。パスが名前ではなくワイルドカード文字で終わる場合は、値 `_1` を使用して行を参照できます。例えば、`SELECT price FROM S3Object[*].books[*].price` の代わりにクエリ `SELECT _1.price FROM S3Object[*].books[*]` を使用できます。
> - Amazon S3 Select は常に JSON ドキュメントをルートレベルの値の配列として扱います。したがって、クエリ対象の JSON オブジェクトにルート要素が 1 つしかない場合でも、FROM 句は `S3Object[*]` で始める必要があります。ただし、互換性の理由から、パスを含めない場合、Amazon S3 Select ではワイルドカード文字を省略できます。そのため、完全な句 `FROM S3Object` は、`FROM S3Object[*] as S3Object` と同等です。パスを含める場合は、ワイルドカード文字も使用する必要があります。つまり、`FROM S3Object` と `FROM S3Object[*].path` はどちらも有効な句ですが、`FROM S3Object.path` は有効な句ではありません。

Example

例:

例 1:

この例では、以下のデータセットとクエリを使用した場合の結果を示しています。

```
{ "Rules": [ {"id": "1"}, {"expr": "y > x"}, {"id": "2", "expr": "z = DEBUG"} ]}
{ "created": "June 27", "modified": "July 6" }
```

```
SELECT id FROM S3Object[*].Rules[*].id
```

```
{"id":"1"}
{}
{"id":"2"}
{}
```

Amazon S3 Select は、以下の理由で次の結果を生成します。

- {"id":"id-1"} — S3Object[0].Rules[0].id は一致を生成しました.
- {} — S3Object[0].Rules[1].id はレコードと一致しなかったため、Amazon S3 Select が MISSING を出力しました。これは、出力のシリアル化中に空のレコードに変更されて返されます。
- {"id":"id-2"} — S3Object[0].Rules[2].id は一致を生成しました。
- {} — S3Object[1] は Rules で一致しなかったため、Amazon S3 Select が MISSING を出力しました。これは、出力のシリアル化中に空のレコードに変更されて返されます。

一致が見つからないときに Amazon S3 Select が空のレコードを返さないようにするには、MISSING という値をテストします。次のクエリは前のクエリと同じ結果を返しますが、空の値は省略されています。

```
SELECT id FROM S3Object[*].Rules[*].id WHERE id IS NOT MISSING
```

```
{"id":"1"}
{"id":"2"}
```

例 2:

この例は、以下のデータセットとクエリを使用した場合の結果を示しています。

```
{ "created": "936864000", "dir_name": "important_docs", "files": [ { "name": "." },
 { "name": ".." }, { "name": ".aws" }, { "name": "downloads" } ], "owner": "Amazon
 S3" }
```

```
{ "created": "936864000", "dir_name": "other_docs", "files": [ { "name": "." },
 { "name": ".." }, { "name": "my stuff" }, { "name": "backup" } ], "owner": "User" }
```

```
SELECT d.dir_name, d.files FROM S3Object[*] d
```

```
{"dir_name":"important_docs","files":[{"name":"."},{"name":".."},{"name":".aws"},
{"name":"downloads"}]}
{"dir_name":"other_docs","files":[{"name":"."},{"name":".."},{"name":"my stuff"},
{"name":"backup"}]}
```

```
SELECT _1.dir_name, _1.owner FROM S3Object[*]
```

```
{"dir_name":"important_docs","owner":"Amazon S3"}
{"dir_name":"other_docs","owner":"User"}
```

WHERE 句

WHERE 句は、この構文に従います。

```
WHERE condition
```

WHERE 句は *condition* に基づいて行をフィルタリングします。condition は、ブール型の結果を持つ式です。condition の結果が TRUE である行のみが結果で返されます。

LIMIT 句

LIMIT 句は、この構文に従います。

```
LIMIT number
```

LIMIT 句により、*number* に基づいてクエリで返されるレコードの数を制限できます。

属性アクセス

SELECT 句および WHERE 句は、クエリ対象のファイルが CSV 形式であるか JSON 形式にあるかによって、以下のセクションのいずれかのメソッドを使用してレコードデータを参照できます。

CSV

- 列番号 – 行の N 番目の列を参照できます。*_N* は列名、N は列位置です。位置カウントは 1 から始まります。たとえば、最初の列名は _1 で、2 番目の列名は _2 です。

  列は *_N* または *alias*.*_N* として参照できます。たとえば、_2 と myAlias._2 はどちらも SELECT リストおよび WHERE 句の列を参照するのに有効な方法です。
- 列ヘッダー – ヘッダー行を持つ CSV 形式のオブジェクトの場合、ヘッダーは SELECT リストおよび WHERE 句で利用できます。特に、従来の SQL では、SELECT および WHERE 句の式内で、*alias*.*column_name* または *column_name* で列を参照できます。

JSON

- ドキュメント – JSON ドキュメントのフィールドは *alias*.*name* としてアクセスできます。ネストされたフィールド (*alias*.*name1*.*name2*.*name3* など) にもアクセスできます。
- リスト – [] 演算子でゼロベースのインデックスを使用して、JSON リストの要素にアクセスできます。例えば、リストの 2 番目の要素に *alias*[1] としてアクセスできます。アクセスするリスト要素とフィールドを組み合わせることができます (例: *alias*.*name1*.*name2*[1].*name3*)。
- 例: この JSON オブジェクトをサンプルデータセットとして考えます。

```
{"name": "Susan Smith",
"org": "engineering",
"projects":
    [
     {"project_name":"project1", "completed":false},
     {"project_name":"project2", "completed":true}
    ]
}
```

例 1:

以下のクエリは次の結果を返します。

```
Select s.name from S3Object s
```

```
{"name":"Susan Smith"}
```

例 2:

以下のクエリは次の結果を返します。

```
Select s.projects[0].project_name from S3Object s
```

```
{"project_name":"project1"}
```

ヘッダーおよび属性名の大文字と小文字の区別

Amazon S3 Select では、二重引用符を使用して、列ヘッダー (CSV オブジェクトの場合) および属性 (JSON オブジェクトの場合) の大文字と小文字を区別することを指定できます。二重引用符がない場合、オブジェクトヘッダーおよび属性の大文字と小文字は区別されません。あいまいな場合はエラーがスローされます。

以下の例は、1) 指定された列ヘッダーを持ち、クエリリクエストに対して `FileHeaderInfo` が `"Use"` に設定されている CSV 形式 の Amazon S3 オブジェクト、または 2) 指定された属性を持つ JSON 形式の Amazon S3 オブジェクトのいずれかです。

例 #1: クエリ対象のオブジェクトには `NAME` というヘッダーまたは属性があります。

- 次の式は、オブジェクトから値を正常に返します。引用符がないため、クエリの大文字と小文字は区別されません。

  ```
  SELECT s.name from S3Object s
  ```

- 次の式は 400 エラー `MissingHeaderName` を返します。引用符があるため、クエリの大文字と小文字は区別されます。

  ```
  SELECT s."name" from S3Object s
  ```

例 #2: クエリ対象の Amazon S3 オブジェクトには `NAME` という 1 つのヘッダーまたは属性と、`name` という別のヘッダーまたは属性があります。

- 次の式は 400 エラー `AmbiguousFieldName` を返します。引用符がないため、クエリの大文字と小文字は区別されませんが、2 つの一致があるため、エラーがスローされます。

  ```
  SELECT s.name from S3Object s
  ```

- 次の式は、オブジェクトから値を正常に返します。引用符があるため、クエリの大文字と小文字が区別されるため、あいまいさはありません。

```
SELECT s."NAME" from S3Object s
```

ユーザー定義の用語としての予約キーワードの使用

Amazon S3 Select には、オブジェクトのコンテンツのクエリに使用する SQL 式を実行するために必要な予約キーワードのセットがあります。予約キーワードには、関数名、データ型、演算子などが含まれます。場合によっては、列ヘッダー (CSV ファイルの場合) や属性 (JSON オブジェクトの場合) などのユーザー定義の用語が、予約キーワードと競合する可能性があります。その場合は、二重引用符を使用して、予約キーワードと競合するユーザー定義の用語を意図的に使用していることを示す必要があります。そうしないと、400 解析エラーが発生します。

予約キーワードの詳細なリストについては、「予約キーワード」を参照してください。

次の例は、1) Amazon S3 オブジェクトが CSV 形式で、指定された列ヘッダーを持ち、クエリリクエストに対して `FileHeaderInfo` が `"Use"` に設定されているか、2) Amazon S3 オブジェクトが JSON 形式で、指定された属性を持っています。

例: クエリ対象のオブジェクトに、予約キーワードである `CAST` という名前のヘッダーまたは属性があります。

- 次の式は、オブジェクトから値を正常に返します。クエリで引用符が使用されているため、S3 Select はユーザー定義のヘッダーまたは属性を使用します。

```
SELECT s."CAST" from S3Object s
```

- 次の式は 400 解析エラーを返します。クエリで引用符が使用されていないため、`CAST` は予約キーワードと競合します。

```
SELECT s.CAST from S3Object s
```

スカラー式

`WHERE` 句および `SELECT` リスト内では、スカラー値を返す式である SQL スカラー式を持つことができます。それには以下の形式があります。

- ***literal***

SQL リテラル。

- ***column_reference***

  形式が *column_name* または *alias*.*column_name* である列への参照。

- ***unary_op expression***

  この場合、***unary_op*** は SQL 単項演算子です。

- ***expression binary_op expression***

  この場合、***binary_op*** は SQL 二項演算子です。

- ***func_name***

  この場合、***func_name*** は、呼び出すスカラー関数の名前です。

- ***expression*** [NOT] BETWEEN ***expression*** AND ***expression***
- ***expression*** LIKE ***expression*** [ESCAPE ***expression***]

データ型

Amazon S3 Select は、複数のプリミティブデータ型をサポートします。

データ型変換

一般的なルールは、CAST 関数 (定義されている場合) に従うことです。CAST が定義されていない場合、すべての入力データが文字列として扱われます。その場合、必要に応じて入力データを関連するデータ型にキャストする必要があります。

CAST 関数の詳細については、「CAST」を参照してください。

サポートされているデータ型

Amazon S3 Select では、プリミティブデータ型の以下のセットがサポートされています。

名前	説明	例
bool	ブール値 (TRUE または FALSE)。	FALSE
int, integer	範囲が -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807 の 8 バイトの符号付き整数。	100000

名前	説明	例
`string`	UTF8 でエンコードされた可変長文字列。デフォルトの制限は 1 文字です。最大文字数制限は 2,147,483,647 文字です。	`'xyz'`
`float`	8 バイトの浮動小数点数。	`CAST(0.456 AS FLOAT)`
`decimal`, `numeric`	10 を底とする数字、最大精度 38 (有効桁の最大数)、かつ -2^{31} から $2^{31}-1$ までの範囲のスケール (10 を底とするべき指数)。 **Note** Amazon S3 Select では、スケールと精度を同時に指定すると、それらは両方とも無視されます。	`123.456`
`timestamp`	タイムスタンプは特定の時点を表し、常にローカルオフセットを含み、任意の精度が可能です。 テキスト形式では、タイムスタンプは日時形式に関する W3C の注記に従いますが、少なくともまる 1 日の精度でなければ、リテラル `T` で終わる必要があります。小数秒は、精度が少なくとも 1 桁また最大は無制限で許可されています。現地時間オフセットは、UTC からの hour:minute オフセットとして、または UTC の現地時刻を示すリテラル `Z` として表すことができます。ローカルタイムオフセットは、時刻のあるタイムスタンプでは必須であり、日付値では許可されていません。	`CAST('2007-04-05T14:30Z' AS TIMESTAMP)`

サポートされている Parquet 型

Amazon S3 Select では、以下の Parquet 型がサポートされています。

- `DATE`
- `DECIMAL`

- ENUM
- INT(8)
- INT(16)
- INT(32)
- INT(64)
- LIST

> **Note**
>
> LIST Parquet 型の出力に関しては、Amazon S3 Select がサポートしているのは JSON 形式のみです。ただし、単純な値のデータに制限してクエリする場合は、CSV 形式で LIST Parquet 型をクエリすることもできます。

- STRING
- TIMESTAMP のサポートされている精度 (MILLIS/MICROS/NANOS)

> **Note**
>
> INT(96) としてのタイムスタンプの保存は、サポートされていません。INT(64) 型の範囲により、NANOS 単位を使用するタイムスタンプのみが、1677-09-21 00:12:43 から 2262-04-11 23:47:16 までの範囲の値を表すことができます。この範囲外の値は、NANOS 単位で表すことはできません。

Amazon S3 Select でサポートされているデータ型への Parquet 型のマッピング

Parquet タイプ	サポートされているデータ型
DATE	timestamp
DECIMAL	decimal, numeric
ENUM	string

Parquet タイプ	サポートされているデータ型
INT(8)	int, integer
INT(16)	int, integer
INT(32)	int, integer
INT(64)	decimal, numeric
LIST	リスト内の各 Parquet 型は、対応するデータ型にマッピングされます。
STRING	string
TIMESTAMP	timestamp

演算子

Amazon S3 Select は、以下の演算子をサポートしています。

論理演算子

- AND
- NOT
- OR

比較演算子

- <
- >
- <=
- >=
- =

- <>
- !=
- BETWEEN
- IN - 例: IN ('a', 'b', 'c')

パターンマッチング演算子

- LIKE
- _ (任意の文字と一致)
- % (任意の一連の文字と一致)

ユニタリー演算子

- IS NULL
- IS NOT NULL

算術演算子

以下に示すとおり、加算、減算、乗算、除算、剰余がサポートされています。

- +
- -
- *
- /
- %

演算子の優先順位

次の表に、演算子の優先順位を高いものから順に示します。

演算子または要素	結合性	必須
-	右	単項マイナス

演算子または要素	結合性	必須
*, /, %	側の	乗算、除算、剰余
+, -	側の	加算、減算
IN		集合要素関係
BETWEEN		範囲指定
LIKE		文字列パターンマッチング
<>		より小さい、より大きい
=	右	等価、割り当て
NOT	右	論理否定
AND	側の	論理積
OR	側の	論理和

予約キーワード

Amazon S3 Select の予約キーワードのリストを以下に示します。これらのキーワードには、オブジェクトのコンテンツをクエリするために使用する SQL 式の実行に必要な関数名、データ型、演算子などが含まれます。

```
absolute
action
add
all
allocate
alter
and
```

```
any
are
as
asc
assertion
at
authorization
avg
bag
begin
between
bit
bit_length
blob
bool
boolean
both
by
cascade
cascaded
case
cast
catalog
char
char_length
character
character_length
check
clob
close
coalesce
collate
collation
column
commit
connect
connection
constraint
constraints
continue
convert
corresponding
count
create
```

```
cross
current
current_date
current_time
current_timestamp
current_user
cursor
date
day
deallocate
dec
decimal
declare
default
deferrable
deferred
delete
desc
describe
descriptor
diagnostics
disconnect
distinct
domain
double
drop
else
end
end-exec
escape
except
exception
exec
execute
exists
external
extract
false
fetch
first
float
for
foreign
found
```

```
from
full
get
global
go
goto
grant
group
having
hour
identity
immediate
in
indicator
initially
inner
input
insensitive
insert
int
integer
intersect
interval
into
is
isolation
join
key
language
last
leading
left
level
like
limit
list
local
lower
match
max
min
minute
missing
module
```

```
month
names
national
natural
nchar
next
no
not
null
nullif
numeric
octet_length
of
on
only
open
option
or
order
outer
output
overlaps
pad
partial
pivot
position
precision
prepare
preserve
primary
prior
privileges
procedure
public
read
real
references
relative
restrict
revoke
right
rollback
rows
schema
```

```
scroll
second
section
select
session
session_user
set
sexp
size
smallint
some
space
sql
sqlcode
sqlerror
sqlstate
string
struct
substring
sum
symbol
system_user
table
temporary
then
time
timestamp
timezone_hour
timezone_minute
to
trailing
transaction
translate
translation
trim
true
tuple
union
unique
unknown
unpivot
update
upper
usage
```

```
user
using
value
values
varchar
varying
view
when
whenever
where
with
work
write
year
zone
```

SQL 関数

Amazon S3 Select は、以下の SQL 関数をサポートしています。

トピック

- 集計関数
- 条件関数
- 変換関数
- 日付関数
- 文字列関数

集計関数

Amazon S3 Select では、以下の集計関数がサポートされています。

関数	引数の型	戻り型
AVG(*expression*)	INT, FLOAT, DECIMAL	INT 引数の場合は DECIMAL、浮動小数点の引数の場合は FLOAT。

関数	引数の型	戻り型
		それ以外の場合は、引数のデータ型と同じです。
COUNT	-	INT
MAX(*expression*)	INT, DECIMAL	引数の型と同じです。
MIN(*expression*)	INT, DECIMAL	引数の型と同じです。
SUM(*expression*)	INT, FLOAT, DOUBLE, DECIMAL	INT 引数の場合は INT、浮動小数点の引数の場合は FLOAT。それ以外の場合は、引数のデータ型と同じです。

SUM の例

S3 インベントリレポート内のフォルダの合計オブジェクトサイズを集計するには、SUM 式を使用します。

次の S3 インベントリレポートは GZIP で圧縮された CSV ファイルです。3 つの列があります。

- 最初の列は、S3 インベントリレポートの対象となる S3 バケットの名前 (*DOC-EXAMPLE-BUCKET*) です。
- 2 番目の列は、バケット内のオブジェクトを一意に識別するオブジェクトキー名です。

  最初の行の *example-folder/* 値は、フォルダ *example-folder* 用です。Amazon S3 で、バケットにフォルダを作成すると、S3 は、指定したフォルダ名に設定されたキーを持つ 0 バイトのオブジェクトを作成します。

2 行目の *example-folder/object1* 値は、*example-folder* フォルダ内のオブジェクト *object1* 用です。

3 行目の *example-folder/object2* 値は、*example-folder* フォルダ内のオブジェクト *object2* 用です。

S3 フォルダの詳細については、「フォルダを使用して Amazon S3 コンソールのオブジェクトを整理する」を参照してください。

- 3 番目の列は、バイト単位のオブジェクトサイズです。

```
"DOC-EXAMPLE-BUCKET","example-folder/","0"
"DOC-EXAMPLE-BUCKET","example-folder/object1","2011267"
"DOC-EXAMPLE-BUCKET","example-folder/object2","1570024"
```

SUM 式を使用してフォルダ *example-folder* の合計サイズを計算するには、Amazon S3 Select で SQL クエリを実行します。

```
SELECT SUM(CAST(_3 as INT)) FROM s3object s WHERE _2 LIKE 'example-folder/%' AND _2 !=
 'example-folder/';
```

クエリ結果:

```
3581291
```

条件関数

Amazon S3 Select は、以下の条件関数をサポートしています。

トピック

- CASE
- COALESCE
- NULLIF

CASE

CASE 式は条件関数であり、他の言語で使われている if/then/else ステートメントと似ています。CASE は、複数の条件がある場合に、結果を指定するために使用します。CASE 式には、簡易と検索の 2 種類があります。

簡易 CASE 式では、式は値と比較されます。一致が検出された場合、THEN 句で指定されたアクションが適用されます。一致が検出されない場合、ELSE 句のアクションが適用されます。

検索 CASE 式では、各 CASE がブール式に基づいて評価され、CASE ステートメントが最初の一致する CASE を返します。一致する CASE が WHEN 句で検出されない場合、ELSE 句のアクションが返されます。

構文

> **Note**
>
> 現時点で、Amazon S3 Select は ORDER BY または新しい行を含むクエリサポートしていません。必ず、改行のないクエリを使用してください。

条件の一致に使用する簡易 CASE ステートメントを以下に示します。

```
CASE expression WHEN value THEN result [WHEN...] [ELSE result] END
```

各条件の評価に使用する検索 CASE ステートメントを以下に示します。

```
CASE WHEN boolean condition THEN result [WHEN ...] [ELSE result] END
```

例

> **Note**
>
> Amazon S3 コンソールを使用して以下の例を実行し、CSV ファイルにヘッダー行が含まれている場合は、[CSV データの最初の行を除外する] を選択します。

例 1: 簡易 CASE 式を使用して、クエリの New York City を Big Apple に置き換えます。その他すべての都市名を other に置換します。

```
SELECT venuecity, CASE venuecity WHEN 'New York City' THEN 'Big Apple' ELSE 'other' END
 FROM S3Object;
```

クエリ結果:

```
venuecity        |   case
-----------------+-----------
Los Angeles      | other
New York City    | Big Apple
San Francisco    | other
Baltimore        | other
...
```

例 2: 検索 CASE 式を使用し、それぞれのチケット販売の pricepaid 値に基づいてグループ番号を割り当てます。

```
SELECT pricepaid, CASE WHEN CAST(pricepaid as FLOAT) < 10000 THEN 'group 1' WHEN
 CAST(pricepaid as FLOAT) > 10000 THEN 'group 2' ELSE 'group 3' END FROM S3Object;
```

クエリ結果:

```
pricepaid |  case
-----------+---------
12624.00 | group 2
10000.00 | group 3
10000.00 | group 3
9996.00 | group 1
9988.00 | group 1
...
```

COALESCE

COALESCE は、引数を順に評価し、最初の不明でない値 (最初の非 Null 値または欠落していない値) を返します。この関数は、Null および欠落している値を伝達しません。

構文

```
COALESCE ( expression, expression, ... )
```

パラメータ

expression

関数の対象となる式。

例

```
COALESCE(1)                -- 1
COALESCE(null)             -- null
COALESCE(null, null)       -- null
COALESCE(missing)          -- null
COALESCE(missing, missing) -- null
COALESCE(1, null)          -- 1
COALESCE(null, null, 1)    -- 1
COALESCE(null, 'string')   -- 'string'
COALESCE(missing, 1)       -- 1
```

NULLIF

2 つの式があり、2 つの式が同じ値に評価される場合、NULLIF は NULL を返し、それ以外の場合、NULLIF は最初の式を評価した結果を返します。

構文

```
NULLIF ( expression1, expression2 )
```

パラメータ

expression1, *expression2*

関数の対象となる式。

例

```
NULLIF(1, 1)             -- null
NULLIF(1, 2)             -- 1
NULLIF(1.0, 1)           -- null
NULLIF(1, '1')           -- 1
NULLIF([1], [1])         -- null
```

```
NULLIF(1, NULL)          -- 1
NULLIF(NULL, 1)          -- null
NULLIF(null, null)       -- null
NULLIF(missing, null)    -- null
NULLIF(missing, missing) -- null
```

変換関数

Amazon S3 Select は、次の変換関数をサポートしています。

トピック

- CAST

CAST

CAST 関数は、エンティティ (1 つの値を返す式など) を 1 つの型から別の型へと変換します。

構文

```
CAST ( expression AS data_type )
```

パラメータ

expression

1 つまたは複数の値、演算子、および値を返す SQL 関数の組み合わせです。

data_type

INT のような、式のキャスト先となる対象データ型です。サポートされているデータ型のリストについては、「データ型」を参照してください。

例

```
CAST('2007-04-05T14:30Z' AS TIMESTAMP)
CAST(0.456 AS FLOAT)
```

日付関数

Amazon S3 Select では、以下の日付関数がサポートされています。

トピック

- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

DATE_ADD

日付部分、数量、タイムスタンプを指定すると、DATE_ADD は日付部分を数量に応じて変更し、更新されたタイムスタンプを返します。

構文

```
DATE_ADD( date_part, quantity, timestamp )
```

パラメータ

date_part

変更する日付の部分を指定します。これには、次のいずれかを指定できます。

- 年
- か月
- 日
- 時間
- 分
- 秒

quantity

更新されたタイムスタンプ値に適用する値。*quantity* が正の値である場合は、タイムスタンプの date_part に加算し、負の値の場合は減算します。

timestamp

関数の対象となるタイムスタンプ。

例

```
DATE_ADD(year, 5, `2010-01-01T`)                -- 2015-01-01 (equivalent to
 2015-01-01T)
DATE_ADD(month, 1, `2010T`)                     -- 2010-02T (result will add precision
 as necessary)
DATE_ADD(month, 13, `2010T`)                    -- 2011-02T
DATE_ADD(day, -1, `2017-01-10T`)                -- 2017-01-09 (equivalent to
 2017-01-09T)
DATE_ADD(hour, 1, `2017T`)                      -- 2017-01-01T01:00-00:00
DATE_ADD(hour, 1, `2017-01-02T03:04Z`)          -- 2017-01-02T04:04Z
DATE_ADD(minute, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:05:05.006Z
DATE_ADD(second, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:04:06.006Z
```

DATE_DIFF

日付部分と 2 つの有効なタイムスタンプを指定すると、DATE_DIFF は日付部分の差を返します。戻り値は、*date_part* の値 *timestamp1* が *date_part* の値 *timestamp2* より大きい場合に負の整数となります。戻り値は、*date_part* の値 *timestamp1* が *date_part* の値 *timestamp2* より少ない場合に正の整数となります。

構文

```
DATE_DIFF( date_part, timestamp1, timestamp2 )
```

パラメータ

date_part

比較するタイムスタンプの部分を指定します。date_part の定義については、「DATE_ADD」を参照してください。

timestamp1

比較する最初のタイムスタンプ。

timestamp2

比較する 2 番目のタイムスタンプ。

例

```
DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`)           -- 1
```

```
DATE_DIFF(year, `2010T`, `2010-05T`)                    -- 4 (2010T is equivalent to
 2010-01-01T00:00:00.000Z)
DATE_DIFF(month, `2010T`, `2011T`)                      -- 12
DATE_DIFF(month, `2011T`, `2010T`)                      -- -12
DATE_DIFF(day, `2010-01-01T23:00`, `2010-01-02T01:00`) -- 0 (need to be at least 24h
 apart to be 1 day apart)
```

EXTRACT

日付部分とタイムスタンプを指定すると、EXTRACT はタイムスタンプの日付部分の値を返します。

構文

```
EXTRACT( date_part FROM timestamp )
```

パラメータ

date_part

抽出するタイムスタンプの部分を指定します。これには、次のいずれかを指定できます。

- YEAR
- MONTH
- DAY
- HOUR
- MINUTE
- SECOND
- TIMEZONE_HOUR
- TIMEZONE_MINUTE

timestamp

関数の対象となるタイムスタンプ。

例

```
EXTRACT(YEAR FROM `2010-01-01T`)                        -- 2010
EXTRACT(MONTH FROM `2010T`)                             -- 1 (equivalent to
 2010-01-01T00:00:00.000Z)
EXTRACT(MONTH FROM `2010-10T`)                          -- 10
```

```
EXTRACT(HOUR FROM `2017-01-02T03:04:05+07:08`)             -- 3
EXTRACT(MINUTE FROM `2017-01-02T03:04:05+07:08`)           -- 4
EXTRACT(TIMEZONE_HOUR FROM `2017-01-02T03:04:05+07:08`)    -- 7
EXTRACT(TIMEZONE_MINUTE FROM `2017-01-02T03:04:05+07:08`)  -- 8
```

TO_STRING

タイムスタンプと形式パターンを指定すると、TO_STRING は、指定された形式でタイムスタンプの文字列表現を返します。

構文

```
TO_STRING ( timestamp time_format_pattern )
```

パラメータ

timestamp

関数の対象となるタイムスタンプ。

time_format_pattern

次の特殊文字の解釈を持つ文字列。

形式	例	説明
yy	69	2 桁の年
y	1969	4 桁の年
yyyy	1969	ゼロ詰めの 4 桁の年
M	1	月
MM	01	ゼロ詰めの月
MMM	Jan	月と年の省略名
MMMM	January	完全な年の名前

形式	例	説明
MMMMM	J	月の最初の文字 (注意: この形式は TO_TIMESTAMP 関数では使用できません)
d	2	日 (1 ~ 31)
dd	02	ゼロ詰めの日 (01 ~ 31)
a	AM	午前または午後
h	3	時間 (1 ~ 12)
hh	03	ゼロ詰めの時間 (01 ~ 12)
H	3	時間 (0 ~ 23)
HH	03	ゼロ詰めの時間 (00 ~ 23)
m	4	分 (0 ~ 59)
mm	04	ゼロ詰めの分 (00 ~ 59)
s	5	秒 (0 ~ 59)
ss	05	ゼロ詰めの秒 (00 ~ 59)

形式	例	説明
S	0	1 秒の端数 (精度: 0.1、範囲: 0.0 ~ 0.9)
SS	6	1 秒の端数 (精度: 0.01、範囲: 0.0 ~ 0.99)
SSS	60	1 秒の端数 (精度: 0.001、範囲: 0.0 ~ 0.999)
…	…	…
SSSSSSSSS	60000000	1 秒の端数 (最大精度: 1 ナノ秒、範囲: 0.0 ~ 0.999999999)
n	60000000	ナノ秒
X	+07 または Z	時間単位のオフセット。オフセットが 0 の場合は Z
XX または XXXX	+0700、または Z	時間および分単位のオフセット。オフセットが 0 の場合は「Z」

形式	例	説明
XXX または XXXXX	+07:00 または Z	時間および分単位のオフセット。オフセットが 0 の場合は Z
x	7	オフセット (時間単位)
xx または xxxx	700	時間および分単位のオフセット
xxx または xxxxx	+07:00	時間および分単位のオフセット

例

```
TO_STRING(`1969-07-20T20:18Z`,  'MMMM d, y')                    -- "July 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMM d, yyyy')                   -- "Jul 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'M-d-yy')                        -- "7-20-69"
TO_STRING(`1969-07-20T20:18Z`, 'MM-d-y')                        -- "07-20-1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y h:m a')               -- "July 20, 1969 8:18
 PM"
TO_STRING(`1969-07-20T20:18Z`, 'y-MM-dd''T''H:m:ssX')           --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00Z`, 'y-MM-dd''T''H:m:ssX')     --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXX')   --
 "1969-07-20T20:18:00+0800"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXXX')  --
 "1969-07-20T20:18:00+08:00"
```

TO_TIMESTAMP

文字列を指定すると、TO_TIMESTAMP は、それをタイムスタンプに変換します。TO_TIMESTAMP は TO_STRING の逆演算です。

構文

```
TO_TIMESTAMP ( string )
```

パラメータ

string

関数の対象となる文字列。

例

```
TO_TIMESTAMP('2007T')                          -- `2007T`
TO_TIMESTAMP('2007-02-23T12:14:33.079-08:00') -- `2007-02-23T12:14:33.079-08:00`
```

UTCNOW

UTCNOW は、現在の時刻をタイムスタンプとして UTC で返します。

構文

```
UTCNOW()
```

パラメータ

UTCNOW はパラメータを受け取りません。

例

```
UTCNOW() -- 2017-10-13T16:02:11.123Z
```

文字列関数

Amazon S3 Select では、以下の文字列関数がサポートされています。

トピック

- CHAR_LENGTH, CHARACTER_LENGTH
- LOWER
- SUBSTRING
- TRIM

- UPPER

CHAR_LENGTH, CHARACTER_LENGTH

CHAR_LENGTH (または CHARACTER_LENGTH) は、指定された文字列の文字数をカウントします。

> **Note**
>
> CHAR_LENGTH と CHARACTER_LENGTH はシノニムです。

構文

```
CHAR_LENGTH ( string )
```

パラメータ

string

関数の対象となる文字列。

例

```
CHAR_LENGTH('')          -- 0
CHAR_LENGTH('abcdefg')   -- 7
```

LOWER

文字列を指定すると、LOWER はすべての大文字を小文字に変換します。大文字以外の文字は変更されません。

構文

```
LOWER ( string )
```

パラメータ

string

関数の対象となる文字列。

例

```
LOWER('AbCdEfG!@#$') -- 'abcdefg!@#$'
```

SUBSTRING

文字列、起動インデックス、長さ (オプション) を指定すると、SUBSTRING は開始インデックスから文字列の末尾、または指定された長さまでの部分文字列を返します。

> **Note**
>
> 入力文字列の先頭文字は、インデックス位置が 1 になります。
>
> - start が 1 未満で、長さを指定しない場合、インデックス位置は 1 に設定されます。
> - start が 1 未満で、長さを指定した場合、インデックス位置は start + length -1 に設定されます。
> - start + length -1 が 0 未満の場合は、空の文字列が返されます。
> - start + length -1 が 0 以上の場合は、インデックス位置 1 から始まる長さ start + length - 1 の部分文字列が返されます。

構文

```
SUBSTRING( string FROM start [ FOR length ] )
```

パラメータ

string

関数の対象となる文字列。

start

文字列の開始位置。

length

返す部分文字列の長さ。存在しない場合は、文字列の末尾に進みます。

例

```
SUBSTRING("123456789", 0)      -- "123456789"
SUBSTRING("123456789", 1)      -- "123456789"
SUBSTRING("123456789", 2)      -- "23456789"
SUBSTRING("123456789", -4)     -- "123456789"
SUBSTRING("123456789", 0, 999) -- "123456789"
SUBSTRING("123456789", 1, 5)   -- "12345"
```

TRIM

文字列の先頭または末尾の文字を切り捨てます。削除するデフォルトの文字はスペース (' ') です。

構文

```
TRIM ( [[LEADING | TRAILING | BOTH remove_chars] FROM] string )
```

パラメータ

string

関数の対象となる文字列。

LEADING | TRAILING | BOTH

このパラメータは、先頭または末尾の文字、またはその両方を切り捨てるかどうかを指定します。

remove_chars

削除する文字のセット。*remove_chars* は、長さが 1 より大きい文字列とすることができます。この関数は、削除された文字列の先頭または末尾で見つかった *remove_chars* から、任意の文字を含む文字列を返します。

例

```
TRIM('       foobar         ')              -- 'foobar'
TRIM('      \tfoobar\t         ')           -- '\tfoobar\t'
TRIM(LEADING FROM '       foobar         ')  -- 'foobar         '
TRIM(TRAILING FROM '       foobar         ') -- '       foobar'
```

```
TRIM(BOTH FROM '       foobar         ')     -- 'foobar'
TRIM(BOTH '12' FROM '1112211foobar22211122') -- 'foobar'
```

UPPER

文字列を指定すると、UPPER はすべての小文字を大文字に変換します。小文字以外の文字は変更されません。

構文

```
UPPER ( string )
```

パラメータ

string

関数の対象となる文字列。

例

```
UPPER('AbCdEfG!@#$') -- 'ABCDEFG!@#$'
```

Amazon S3 オブジェクトでの大規模なバッチ操作の実行

S3 バッチ操作を使用すると、Amazon S3 のオブジェクトに対して大規模なバッチ操作を実行することができます。S3 バッチ操作では、指定した Amazon S3 のオブジェクトのリストに対して、1 つのオペレーションを実行できます。1 つのジョブで、エクサバイトのデータを含む数十億ものオブジェクトに対して、指定されたオペレーションを実行できます。Amazon S3 は、進捗状況の追跡、通知の送信、すべてのアクションの詳細な完了レポートの保存を行い、フルマネージド型の監査可能なサーバーレスエクスペリエンスを提供します。AWS Management Console、AWS CLI、Amazon SDK、または REST API を使用して S3 バッチ操作を使用できます。

S3 バッチ操作は、オブジェクトをコピーしたり、オブジェクトにタグやアクセスコントロールリスト (ACL) を設定したりするために使用できます。また、S3 Glacier Flexible Retrieval からオブジェクトの復元を行ったり、AWS Lambda 関数を呼び出してオブジェクトを使用してカスタムアクションを実行したりすることもできます。これらのオペレーションは、指定したオブジェクトのカスタムリストに対して実行できます。また、Amazon S3 インベントリレポートを使用して、オブジェクトのリストを簡単に作成することもできます。Amazon S3 バッチ操作では、Amazon S3 で既に使用

しているものと同じ Amazon S3 の API を使用するため、使い慣れたインターフェイスを使用できます。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。S3 Express One Zone とディレクトリバケットでのバッチオペレーションの使用の詳細については、「S3 Express One Zone でのバッチオペレーションの使用」を参照してください。

S3 バッチ操作の基本

S3 バッチ操作を使用すると、Amazon S3 のオブジェクトに対して大規模なバッチ操作を実行することができます。S3 バッチ操作では、指定した Amazon S3 のオブジェクトのリストに対して、1 つのオペレーションまたはアクションを実行できます。

用語

このセクションでは、次のように定義されるジョブ、オペレーション、およびタスクの用語を使用します。

ジョブ

ジョブは、S3 バッチ操作の基本単位です。ジョブには、マニフェストにリストされているオブジェクトに対して指定されたオペレーションを実行するために必要なすべての情報が含まれています。この情報を入力してからジョブの開始をリクエストすると、ジョブはマニフェスト内の各オブジェクトにおいてオペレーションを実行します。

オペレーション

オペレーションは、バッチ操作のジョブで実行する API のアクションの種類 (オブジェクトのコピーなど) です。各ジョブは、マニフェストで指定されているすべてのオブジェクトに対して 1 種類のオペレーションを実行します。

タスク

タスクは、ジョブ実行の単位です。タスクは、1 つのオブジェクトに対してジョブのオペレーションを実行するための Amazon S3 または AWS Lambda API のオペレーションの 1 回の呼び出

しを表します。S3 バッチ操作は、ジョブの実行中に、マニフェストで指定されている各オブジェクトに対して 1 つのタスクを作成します。

S3 バッチ操作のジョブの仕組み

ジョブは、S3 バッチ操作の基本単位です。ジョブには、オブジェクトのリストに対して指定されたオペレーションを実行するために必要なすべての情報が含まれています。ジョブを作成するには、S3 バッチ操作にオブジェクトのリストを渡し、それらのオブジェクトに対して実行するアクションを指定します。

S3 バッチ操作がサポートするオペレーションの詳細については、S3 バッチ操作でサポートされるオペレーション を参照してください。

バッチジョブは、そのマニフェストに含まれるすべてのオブジェクトで指定されたオペレーションを実行します。マニフェストには、バッチジョブで処理するオブジェクトが一覧表示され、オブジェクトとしてバケットに保存されます。カンマ区切り値 (CSV) 形式の Amazon S3 インベントリ レポートをマニフェストとして使用できます。これにより、バケット内に配置されたオブジェクトの大きなリストを簡単に作成できます。1 つのバケット内に含まれるカスタマイズされたオブジェクトのリストに対してバッチ操作を実行できるように、シンプルな CSV 形式でマニフェストを指定することもできます。

ジョブを作成すると、Amazon S3 はマニフェストにリストされているオブジェクトを処理し、指定されたオペレーションをそれぞれのオブジェクトに対して実行します。ジョブの実行中は、プログラムまたは Amazon S3 コンソールで進捗状況をモニタリングできます。終了時に完了レポートを生成するようにジョブを設定することもできます。完了レポートには、ジョブによって実行された各タスクの結果が示されます。ジョブのモニタリングの詳細については、「S3 バッチ操作ジョブの管理」を参照してください。

S3 バッチ操作のチュートリアル

次のチュートリアルでは、いくつかのバッチ操作タスクにおけるエンドツーエンドの一連の手順について説明します。

- チュートリアル: S3 バッチオペレーション、AWS Lambda、および AWS Elemental MediaConvert を使用した動画のバッチトランスコーディング

Amazon S3 バッチ操作に対するアクセス許可の付与

S3 バッチ操作ジョブを作成して実行する前に、必要な許可を付与する必要があります。Amazon S3 バッチ操作ジョブを作成するには、`s3:CreateJob` ユーザー許可が必要です。ジョブを作成する同じエンティティには、ジョブに対して指定された AWS Identity and Access Management (IAM) ロールをバッチ操作に渡すための `iam:PassRole` 許可も必要です。

IAM リソースの指定に関する一般的な情報については、IAM ユーザーガイドの IAM JSON ポリシー、リソース要素を参照してください。以下のセクションでは、IAM ロールの作成とポリシーのアタッチについて説明します。

トピック

- S3 バッチ操作の IAM ロールの作成
- 許可ポリシーのアタッチ

S3 バッチ操作の IAM ロールの作成

Amazon S3 には、ユーザーに代わって S3 バッチ操作を実行するための許可が必要です。AWS Identity and Access Management (IAM) ロールを介してこれらのアクセス許可を付与します。このセクションでは、IAM ロールを作成するときに使用する信頼ポリシーとアクセス許可ポリシーの例を示します。詳細については、IAM ユーザーガイドの IAM ロールを参照してください。例については、ジョブタグを使用した S3 バッチ操作のアクセス許可の制御 および S3 バッチ操作を使用したオブジェクトのコピー を参照してください。

IAM ポリシーでは、条件キーを使用して、S3 バッチ操作ジョブのアクセス許可をフィルタリングすることもできます。Amazon S3 固有の条件キーの詳細な情報と完全なリストについては、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

信頼ポリシー

S3 バッチ操作のサービスプリンシパルが IAM ロールを引き受けることを許可するには、ロールに次の信頼ポリシーをアタッチします。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
```

```
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

許可ポリシーのアタッチ

オペレーションのタイプに応じて、以下のいずれかのポリシーをアタッチできます。

アクセス権限を設定する前に、以下の点にご留意ください。

- オペレーションに関係なく、Amazon S3 には S3 バケットからマニフェストオブジェクトを読み込み、またオプションでバケットにレポートを書き込むアクセス許可が必要です。そのため、次のポリシーにはすべて、これらのアクセス許可が含まれます。
- Amazon S3 インベントリレポートマニフェストの場合、S3 バッチ操作では manifest.json オブジェクトおよび関連するすべての CSV データファイルを読み込むためのアクセス許可が必要です。
- オブジェクトのバージョン ID を指定している場合にのみ、`s3:GetObjectVersion` などのバージョン固有のアクセス許可が必要です。
- 暗号化されたオブジェクトに対して S3 バッチ操作を実行している場合、IAM ロールには、暗号化に使用される AWS KMS キーへのアクセス許可も必要です。
- インベントリレポートマニフェストを AWS KMS で暗号化して送信する場合、IAM ポリシーでは manifest.json オブジェクトおよび関連するすべての CSV データファイルに対するアクセス許可の `"kms:Decrypt"` と `"kms:GenerateDataKey"` が必要です。
- バッチオペレーションジョブにより、ACL が有効で、別の AWS アカウントのバケットにマニフェストを生成する場合は、そのバッチジョブ用に設定された IAM ロールの IAM ポリシーで `s3:PutObjectAcl` 権限を付与する必要があります。この権限を含めない場合、バッチジョブは `Error occurred when preparing manifest: Failed to write manifest` エラーで失敗します。

オブジェクトをコピー: PutObject

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectTagging"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DestinationBucket/*"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::SourceBucket",
                "arn:aws:s3:::SourceBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

オブジェクトタグを置換: PutObjectTagging

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::TargetResource/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ManifestBucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::ReportBucket/*"
      ]
    }
  ]
}
```

オブジェクトタグを削除: DeleteObjectTagging

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:DeleteObjectTagging",
                "s3:DeleteObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::TargetResource/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

アクセスコントロールリストを置換: PutObjectAcl

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl"
      ],
      "Resource": "arn:aws:s3:::TargetResource/*"
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ManifestBucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::ReportBucket/*"
      ]
    }
  ]
}
```

オブジェクトを復元: RestoreObject

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
          "s3:RestoreObject"
      ],
      "Resource": "arn:aws:s3:::TargetResource/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ManifestBucket/*"
      ]
    },
```

```
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::ReportBucket/*"
      ]
    }
  ]
}
```

オブジェクトロックの保持を適用: PutObjectRetention

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::TargetResource"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectRetention",
                "s3:BypassGovernanceRetention"
            ],
            "Resource": [
                "arn:aws:s3:::TargetResource/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

オブジェクトロックのリーガルホールドを適用: PutObjectLegalHold

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::TargetResource"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "s3:PutObjectLegalHold",
            "Resource": [
                "arn:aws:s3:::TargetResource/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

既存の objects: InitiateReplication を S3 で生成されたマニフェストでレプリケートする

S3 で生成されたマニフェストを使用して保存する場合にはこのポリシーを使用します。既存のオブジェクトのバッチオペレーションの詳細については、「S3 バッチレプリケーションを使用した既存のオブジェクトのレプリケーション」を参照してください。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
         ]
      },
      {
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:PutInventoryConfiguration"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***"
         ]
      },
      {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
```

```
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::*** completion report bucket ****/*",
            "arn:aws:s3:::*** manifest bucket ****/*"
         ]
      }
   ]
}
```

既存の objects: InitiateReplication をユーザーマニフェストでレプリケートする

ユーザー指定のマニフェストを使用する場合にはこののポリシーを使用します。既存のオブジェクトのバッチオペレーションの詳細については、「S3 バッチレプリケーションを使用した既存のオブジェクトのレプリケーション」を参照してください。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
         ]
      },
            {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
```

```
            "Resource":[
               "arn:aws:s3:::*** manifest bucket ***/*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:PutObject"
            ],
            "Resource":[
               "arn:aws:s3:::*** completion report bucket ****/*"
            ]
         }
      ]
}
```

S3 バッチオペレーションジョブの作成

Amazon S3 バッチオペレーションを使用すると、特定の Amazon S3 オブジェクトのリストに対して大規模なバッチオペレーションを実行できます。このセクションでは、S3 バッチ操作ジョブの作成に必要な情報と `CreateJob` リクエストの結果について説明します。また、AWS Command Line Interface (AWS CLI)、AWS SDK for Java を使用したバッチオペレーションジョブの作成手順についても説明します。

S3 バッチオペレーションジョブを作成するときに、すべてのタスクまたは失敗したタスクについてのみ完了レポートをリクエストできます。少なくとも 1 つのタスクが正常に呼び出される限り、S3 バッチオペレーションは、完了、失敗、またはキャンセルされたジョブに関するレポートを生成します。詳細については、「例: S3 バッチ操作完了レポート」を参照してください。

トピック

- バッチ操作ジョブのリクエストの要素
- マニフェストの指定

バッチ操作ジョブのリクエストの要素

S3 バッチ操作ジョブを作成するには、次の情報を指定する必要があります。

オペレーション

S3 バッチ操作でマニフェストのオブジェクトに対して実行するオペレーションを指定します。オペレーションのタイプごとに、そのオペレーションに固有のパラメータを受け入れます。バッチオペレーションを使用すると、オペレーションを一括で実行でき、各オブジェクトに対してそのオペレーションを 1 つずつ実行した場合と同じ結果が得られます。

マニフェスト

マニフェストは、S3 バッチオペレーションで指定したアクションを実行するすべてのオブジェクトのリストです。バッチオペレーションジョブのマニフェストは、次の方法で指定できます。

- カスタマイズした CSV 形式の独自のオブジェクトリストを手動で作成します。
- 既存の CSV-形式の Amazon S3 インベントリ レポートを選択します。
- 直接バッチオペレーションでは、ジョブの作成時に指定したオブジェクトフィルター条件に基づいてマニフェストを自動的に生成します。このオプションは、Amazon S3 コンソールで作成したバッチレプリケーションジョブ、AWS CLI、AWS SDK、または Amazon S3 REST API を使用して作成した任意のジョブタイプで使用できます。

> **Note**
>
> - マニフェストの指定方法を問わず、リスト自体は汎用バケットに保存する必要があります。バッチオペレーションでは、既存のマニフェストをディレクトリバケットからインポートしたり、生成されたマニフェストをディレクトリバケットに保存したりすることはできません。ただし、マニフェスト内に記述されたオブジェクトはディレクトリバケットに保存できます。詳細については、「ディレクトリバケット」を参照してください。
> - マニフェスト内のオブジェクトがバージョニング対象のバケット内にある場合は、そのオブジェクトのバージョン ID を指定すると特定のバージョンに対してオペレーションが実行されます。バージョン ID が指定されていない場合、バッチオペレーションはオブジェクトの最新バージョンに対してオペレーションを実行します。マニフェストにバージョン ID フィールドが含まれている場合は、マニフェスト内にあるすべてのオブジェクトのバージョン ID を指定する必要があります。

詳細については、「マニフェストの指定」を参照してください。

優先度

ジョブの優先順位を使用して、自分のアカウントで実行中の他のユーザーに対するこのジョブの相対的な優先順位を示します。番号が大きいほど、優先度が高いことを表します。

ジョブの優先度は、同じアカウントとリージョンの他のジョブに設定された優先度に相対して解釈されます。お客様に合った番号付け体系を選択できます。例えば、すべての [復元] (`RestoreObject`) ジョブに優先度 1、すべての [コピー] (`CopyObject`) ジョブに優先度 2、すべての [アクセスコントロールリスト (ACL) に置き換える] (`PutObjectAcl`) ジョブに優先度 3 を割り当てることができます。

S3 バッチオペレーションでは、優先度の数字に従ってジョブに優先順位が付けられます。ただし、厳密に順序付けされるとは限りません。そのため、いずれかのジョブをその他のジョブよりも前に開始または終了させるためにジョブの優先度は使用すべきではありません。厳密な順序付けする必要がある場合は、1 つのジョブが終了するまで待ってから次のジョブを開始します。

RoleArn

ジョブを実行する AWS Identity and Access Management (IAM) ロールを指定します。使用する IAM ロールには、そのジョブで指定されているオペレーションを実行するための十分なアクセス許可が必要です。例えば、`CopyObject` ジョブを実行するには、IAM ロールに、ソースバケットに対する `s3:GetObject` アクセス許可と、送信先バケットに対する `s3:PutObject` アクセス許可が必要です。このロールには、マニフェストを読み取り、ジョブ完了レポートを書き込むためのアクセス許可も必要です。

IAM ロールの詳細については、「IAM ユーザーガイド」の「IAM ロール」を参照してください。

Amazon S3 のアクセス許可の詳細については、「Amazon S3 のポリシーアクション」を参照してください。

> **Note**
>
> ディレクトリバケットに対してアクションを実行するバッチオペレーションジョブには、特定のアクセス許可が必要です。詳細については、S3 Express One Zone 向け AWS Identity and Access Management (IAM)」を参照してください。

レポートを行う

S3 バッチ操作で完了レポートを生成するかどうかを指定します。ジョブ完了レポートをリクエストする場合、この要素にレポートのパラメータも指定する必要があります。必要な情報には以下が含まれます。

- レポートを保存するバケット。

> **Note**
>
> レポートは汎用バケットに保存する必要があります。バッチオペレーションではレポートをディレクトリバケットに保存できません。詳細については、「ディレクトリバケット」を参照してください。

- レポートの形式。
- レポートに、すべてのタスクの詳細を含めるか、失敗したタスクのみを含めるかを指定します。
- オプションのプレフィックス文字列

> **Note**
>
> 完了レポートは常に、Amazon S3 マネージドキー (SSE-S3) で暗号化されます。

[タグ (省略可能)]

タグを追加することで、S3 バッチ操作ジョブへのラベル付けとアクセスの制御を実行できます。タグを使用して、バッチオペレーションジョブの担当者を識別したり、ユーザーがバッチオペレーションジョブを操作する方法を制御したりできます。ジョブタグがあることで、ユーザーによるジョブのキャンセル、確認状態にあるジョブの有効化、ジョブの優先度レベルの変更を許可したり制限したりできます。例えば、ジョブが `"Department=Finance"` タグ付きで作成されている場合、`CreateJob` オペレーションを呼び出すアクセス許可をユーザーに付与できます。

タグをアタッチしてジョブを作成し、後でジョブにタグを追加できます。

詳細については、「the section called “タグの使用”」を参照してください。

Description (オプション)

ジョブを追跡および監視するために、最大 256 文字の説明を指定することもできます。Amazon S3 では、ジョブに関する情報を返すか、Amazon S3 コンソールにジョブの詳細を表示するたび

に、この説明が含まれます。これによって、入力した説明に応じて簡単にジョブを並べ替えたりフィルタリングしたりできます。説明は一意である必要はないので、類似したジョブのグループを追跡するのに役立つように、説明をカテゴリとして使用することができます (「Weekly Log Copy Jobs」など)。

マニフェストの指定

マニフェストは、Amazon S3 が動作するオブジェクトキーを含む Amazon S3 オブジェクトです。次のいずれかの方法でマニフェストを提供できます。

- 新しいマニフェストファイルを手動で作成する。
- 既存のマニフェストを使用する。
- 直接バッチオペレーションでは、ジョブの作成時に指定したオブジェクトフィルター条件に基づいてマニフェストを自動的に生成します。このオプションは、Amazon S3 コンソールで作成したバッチレプリケーションジョブ、AWS CLI、AWS SDK、または Amazon S3 REST API を使用して作成した任意のジョブタイプで使用できます。

> **Note**
>
> マニフェストの指定方法を問わず、リスト自体は汎用バケットに保存する必要があります。バッチオペレーションでは、既存のマニフェストをディレクトリバケットからインポートしたり、生成されたマニフェストをディレクトリバケットに保存したりすることはできません。ただし、マニフェスト内に記述されたオブジェクトはディレクトリバケットに保存できます。詳細については、「ディレクトリバケット」を参照してください。

マニフェストファイルの作成

ジョブのマニフェストを手動で作成するには、マニフェストオブジェクトキー、ETag (エンティティタグ)、オプションでバージョン ID を CSV 形式のリストで指定します。マニフェストの内容は URL エンコードされている必要があります。

デフォルトでは、Amazon S3 は、Amazon S3 マネージドキー (SSE-S3) を使用したサーバー側暗号化を自動的に使用して、Amazon S3 バケットにアップロードされるマニフェストを暗号化します。顧客指定のキーを使用したサーバー側の暗号化 (SSE-C) を使用するマニフェストはサポートされていません。AWS Key Management Service (AWS KMS) キー (SSE-KMS) でサーバー側の暗号化を

使用するマニフェストは、CSV 形式のインベントリレポートを使用する場合にのみサポートされます。AWS KMS を使用したマニフェストの手動作成はサポートされていません。

マニフェストには、各オブジェクトのバケット名、オブジェクトキー、およびオプションでオブジェクトバージョンを含める必要があります。マニフェストのその他のフィールドは、S3 バッチ操作では使用されません。

> **Note**
>
> マニフェスト内のオブジェクトがバージョニング対象のバケット内にある場合は、そのオブジェクトのバージョン ID を指定すると特定のバージョンに対してオペレーションが実行されます。バージョン ID が指定されていない場合、バッチオペレーションはオブジェクトの最新バージョンに対してオペレーションを実行します。マニフェストにバージョン ID フィールドが含まれている場合は、マニフェスト内にあるすべてのオブジェクトのバージョン ID を指定する必要があります。

CSV 形式のマニフェスト (バージョン ID なし) の例を以下に示します。

```
Examplebucket,objectkey1
Examplebucket,objectkey2
Examplebucket,objectkey3
Examplebucket,photos/jpgs/objectkey4
Examplebucket,photos/jpgs/newjersey/objectkey5
Examplebucket,object%20key%20with%20spaces
```

バージョン ID を含む CSV 形式のマニフェストの例は次のとおりです。

```
Examplebucket,objectkey1,PZ9ibn9D5lP6p298B7S9_ceqx1n5EJ0p
Examplebucket,objectkey2,YY_ouuAJByNW1LRBfFMfxMge7XQWxMBF
Examplebucket,objectkey3,jbo9_jhdPEyB4RrmOxWS0kU0EoNrU_oI
Examplebucket,photos/jpgs/objectkey4,6EqlikJJxLTsHsnbZbSRffn24_eh5Ny4
Examplebucket,photos/jpgs/newjersey/objectkey5,imHf3FAiRsvBW_EHB8GOu.NHunHO1gVs
Examplebucket,object%20key%20with%20spaces,9HkPvDaZY5MVbMhn6TMn1YTb5ArQAo3w
```

既存のマニフェストファイルの指定

次の 2 つの形式のいずれかを使用して、ジョブ作成リクエストのマニフェストファイルを指定できます。

- Amazon S3 インベントリレポート - CSV 形式の Amazon S3 インベントリレポートである必要があります。インベントリレポートに関連付けられた `manifest.json` ファイルを指定する必要があります。インベントリレポートの詳細については、「Amazon S3 インベントリ」を参照してください。インベントリレポートにバージョン ID が含まれている場合、S3 バッチ操作は特定のオブジェクトのバージョンに対して実行されます。

  > **Note**
  >
  > - S3 バッチオペレーションは、SSE-KMS で暗号化された CSV インベントリレポートをサポートしています。
  > - SSE-KMS で暗号化されたインベントリレポートマニフェストを送信する場合、IAM ポリシーには、`"kms:Decrypt"`、`manifest.json` オブジェクトのための `"kms:GenerateDataKey"`、関連するすべての CSV データ ファイルに対するアクセス許可が含まれている必要があります。

- CSV ファイル – ファイルの各行には、バケット名とオブジェクトのキーを含める必要があります。また、任意でオブジェクトのバージョンを含めることができます。オブジェクトキーは、次の例に示されているように、URL エンコードする必要があります。マニフェストには、すべてのオブジェクトのバージョン ID を含めるか、すべてのオブジェクトのバージョン ID を省略する必要があります。CSV マニフェスト形式の詳細については、「Amazon Simple Storage Service API リファレンス」の「JobManifestSpec」を参照してください。.

  > **Note**
  >
  > S3 バッチオペレーションは、SSE-KMS で暗号化された CSV マニフェストファイルをサポートしていません。

> **Important**
>
> 手動で作成したマニフェストとバージョニング対応のバケットを使用する場合は、そのオブジェクトのバージョン ID を指定することをお勧めします。ジョブを作成すると、S3 バッチ操作はジョブを実行する前にマニフェスト全体を解析します。ただし、これによってバケットの状態は「スナップショット」されません。
> マニフェストには数十億のオブジェクトが含まれる可能性があるため、ジョブの実行に時間がかかる可能性があり、その結果ジョブが動作するオブジェクトのバージョンに影響を与える場合があります。ジョブの実行中にオブジェクトを新しいバージョンで上書きして、その

オブジェクトのバージョン ID を指定しなかったとします。この場合、Amazon S3 は、ジョブの作成時に存在していたバージョンではなく、オブジェクトの最新バージョンに対してオペレーションを実行します。この動作を回避する唯一の方法は、マニフェスト内でリストされたオブジェクトにバージョン ID を指定することです。

マニフェストの自動的生成

ジョブの作成時に指定したオブジェクトフィルター条件に基づいてマニフェストを自動的に生成するように Amazon S3 に指示できます。このオプションは、Amazon S3 コンソールで作成したバッチレプリケーションジョブ、AWS CLI、AWS SDK、または Amazon S3 REST API を使用して作成した任意のジョブタイプで使用できます。バッチレプリケーションの詳細については、「S3 バッチレプリケーションを使用した既存のオブジェクトのレプリケーション」を参照してください。

マニフェストを自動的に生成するには、ジョブ作成リクエストの一環として次の要素を指定します。

- バケット所有者や Amazon リソースネーム (ARN) など、ソースオブジェクトが含まれるバケットに関する情報
- マニフェストファイルを作成するためのフラグ、出力バケット所有者、ARN、プレフィックス、ファイル形式、暗号化タイプなど、マニフェスト出力に関する情報
- 作成日、キー名、サイズ、ストレージクラス、タグでオブジェクトをフィルタリングするためのオプションの条件

オブジェクトフィルターの条件

自動生成されたマニフェストに含まれるオブジェクトのリストをフィルタリングするには、次の条件を指定できます。詳細については、「Amazon S3 API リファレンス」の「JobManifestGeneratorFilter」を参照してください。

CreatedAfter

指定した場合、生成されたマニフェストには、この時間以降に作成されたソースバケットオブジェクトのみが含まれます。

CreatedBefore

指定した場合、生成されたマニフェストには、この時間以降に作成されたソースバケットオブジェクトのみが含まれます。

EligibleForReplication

指定した場合、生成されたマニフェストには、ソースバケットのレプリケーション設定に従ってレプリケーションの対象となるオブジェクトのみが含まれます。

KeyNameConstraint

指定された場合、生成されたマニフェストには、オブジェクトキーが MatchAnySubstring、MatchAnyPrefix、MatchAnySuffix に指定された文字列制約と一致するソースバケットオブジェクトのみが含まれます。

MatchAnySubstring – 指定した場合、指定された文字列がオブジェクトキー文字列内のいずれかの部分に出現する場合、生成されたマニフェストにそのオブジェクトが含まれます。

MatchAnyPrefix – 指定すると、指定された文字列がオブジェクトキー文字列の先頭に現れる場合、生成されたマニフェストにオブジェクトが含まれます。

MatchAnySuffix – 指定した場合、指定された文字列がオブジェクトキー文字列の末尾に出現する場合、生成されたマニフェストにオブジェクトが含まれます。

MatchAnyStorageClass

指定した場合、生成されたマニフェストには、指定されたストレージクラスで保存されているソースバケットオブジェクトのみが含まれます。

ObjectReplicationStatuses

指定した場合、生成されたマニフェストには、指定されたレプリケーションステータスのいずれかを持つソースバケットオブジェクトのみが含まれます。

ObjectSizeGreaterThanBytes

指定した場合、生成されたマニフェストには、ファイルサイズが指定されたバイト数以上のソースバケットオブジェクトのみが含まれます。

ObjectSizeLessThanBytes

指定した場合、生成されたマニフェストには、ファイルサイズが指定されたバイト数未満のソースバケット オブジェクトのみが含まれます。

> **Note**
>
> マニフェストが自動的に生成されたほとんどのジョブはクローンの作成はできません。バッチ レプリケーション ジョブ

はKeyNameConstraint、MatchAnyStorageClass、ObjectSizeGreaterThanBytes、または ObjectSizeLessThanBytes マニフェストフィルター条件を使用する場合を除き、クローンを作成できます。

マニフェスト条件を指定する構文は、ジョブの作成に使用する方法に応じて異なります。例については、「ジョブの作成」を参照してください。

ジョブの作成

Amazon S3 コンソール、AWS CLI、AWS SDK、または Amazon S3 REST API を使用して S3 バッチオペレーションジョブを作成できます。

ジョブリクエストの作成の詳細については、バッチ操作ジョブのリクエストの要素 を参照してください。

前提条件

バッチオペレーションジョブを作成する前に、関連するアクセス許可が設定されていることを確認します。詳細については、「Amazon S3 バッチ操作に対するアクセス許可の付与」を参照してください。

S3 コンソールの使用

バッチジョブを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ページ上部にあるナビゲーションバーで、現在表示されている AWS リージョン の名前をクリックします。次に、ジョブを作成するリージョンを選択します。

   > **Note**
   >
   > コピーオペレーションの場合は、コピー先バケットと同じリージョンにジョブを作成する必要があります。その他すべてのオペレーションの場合は、マニフェスト内のオブジェクトと同じリージョンにジョブを作成する必要があります。

3. Amazon S3 コンソールのナビゲーションペインで [バッチオペレーション] を選択します。
4. [ジョブの作成] を選択します。
5. ジョブを作成する AWS リージョン を確認します。

6. [マニフェストの形式] で、使用するマニフェストオブジェクトのタイプを選択します。

   - [S3 inventory report (S3 インベントリレポート)] を選択する場合、CSV 形式のインベントリレポートの一部として Amazon S3 が生成する manifest.json オブジェクトへのパスと、最新のバージョンではなく特定のバージョンを使用する場合には、オプションでマニフェストオブジェクトのバージョン ID を入力します。
   - [CSV] を選択する場合は、CSV 形式のマニフェストオブジェクトへのパスを入力します。マニフェストオブジェクトは、コンソールで説明される形式に従う必要があります。最新のバージョンではなく特定のバージョンを使用する場合には、マニフェストオブジェクトのバージョン ID をオプションで含めることもできます。

   > **Note**
   >
   > Amazon S3 コンソールは、バッチレプリケーションジョブのみの自動マニフェスト生成をサポートします。その他のすべてのジョブタイプで、指定したフィルタ条件に基づいて Amazon S3 がマニフェストを自動的に生成するようにするには、AWS CLI、AWS SDK、または Amazon S3 REST API を使用してジョブを設定する必要があります。

7. [Next (次へ)] を選択します。
8. [Operation (オペレーション)] で、マニフェストに登録されたすべてのオブジェクトを実行するオペレーションを選択します。選択したオペレーションに関する情報を入力して、[次へ] を選択します。
9. [追加オプションの設定] に関する情報を入力して、[次へ] を選択します。
10. [Review (確認)] で、設定を確認します。変更が必要な場合は、[戻る] を選択します。それ以外の場合は、[ジョブの作成] を選択します。

AWS CLI の使用

Specify manifest

既存のマニフェスト ファイルにリストされているオブジェクトに対して動作する S3 バッチオペレーション `S3PutObjectTagging` ジョブを作成する方法は、次の例のとおりです。

バッチ操作 **`S3PutObjectTagging`** ジョブを作成するには

1. 次のコマンドを使用して AWS Identity and Access Management (IAM) ロールを作成してから、IAM ポリシーを作成して関連するアクセス許可を割り当てます。次のロールとポリシー

は、オブジェクト タグを追加するための Amazon S3 アクセス許可を付与します。このアクセス許可は、以降のステップでジョブを作成する際に必要になります。

a. 次のコマンド例を使用して、バッチオペレーションで使用する IAM ロールを作成します。このコマンド例を使用するには、*S3BatchJobRole* をロールにつける名前に置き換えます。

```
aws iam create-role \
 --role-name S3BatchJobRole \
 --assume-role-policy-document '{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}'
```

ロールの Amazon リソースネーム (ARN) を記録します。ジョブの作成時にこの ARN が必要となります。

b. 次のコマンド例を使用して、必要なアクセス許可を持つ IAM ポリシーを作成して、前の手順で作成した IAM ロールにアタッチします。必要なアクセス権限の詳細については、「Amazon S3 バッチ操作に対するアクセス許可の付与」を参照してください。

> **Note**
>
> ディレクトリバケットに対してアクションを実行するバッチオペレーションジョブには、特定のアクセス許可が必要です。詳細については、S3 Express One Zone 向け AWS Identity and Access Management (IAM)」を参照してください。

このサンプルコマンドを使用するには、*user input placeholders* を次のとおり置き換えます。

- *S3BatchJobRole* は、IAM ロール名に置き換えます。この名前が以前に使用した名前と一致することを確認します。
- *PutObjectTaggingBatchJobPolicy* は、IAM ポリシーにつける名前に置き換えます。
- *example-s3-destination-bucket* は、タグを付けるオブジェクトがあるバケット名と置き換えます。
- *DOC-EXAMPLE-MANIFEST-BUCKET* は、マニフェストがあるバケットの名前と置き換えます。
- *DOC-EXAMPLE-REPORT-BUCKET* は、完了レポートの配信先のバケット名と置き換えます。

```
aws iam put-role-policy \
  --role-name S3BatchJobRole \
  --policy-name PutObjectTaggingBatchJobPolicy \
  --policy-document '{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::example-s3-destination-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-BUCKET/*"
      ]
    },
    {
```

```
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:GetBucketLocation"
      ],
      "Resource":[
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET/*"
      ]
    }
  ]
}'
```

2. 次のコマンド例を使用して、S3PutObjectTagging ジョブを作成します。

   manifest.csv ファイルは、バケットのリストとオブジェクトキー値を提供します。このジョブは、マニフェストで識別されたオブジェクトに指定されたタグを適用します。ETag は manifest.csv オブジェクトの ETag であり、Amazon S3 コンソールから取得できます。このリクエストは no-confirmation-required パラメータを指定するため、update-job-status コマンドで確認しなくてもジョブを実行できます。詳細については、AWS CLI コマンドリファレンスの「create-job」を参照してください。

   このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。*IAM-role* を先ほど作成した IAM ロールの ARN に置き換えます。

```
aws s3control create-job \
    --region us-west-2 \
    --account-id acct-id \
    --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"keyOne",
 "Value":"ValueOne"}] }}' \
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":
{"ObjectArn":"arn:aws:s3:::my_manifests/
manifest.csv","ETag":"60e460c9d1046e73f7dde5043ac3ae85"}}' \
    --report '{"Bucket":"arn:aws:s3:::DOC-EXAMPLE-REPORT-
BUCKET","Prefix":"final-reports",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
    --priority 42 \
    --role-arn IAM-role \
    --client-request-token $(uuidgen) \
    --description "job description" \
    --no-confirmation-required
```

応答として、Amazon S3 はジョブ ID (など `00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c`) を返します。ジョブの識別、モニタリング、変更にはこのジョブ ID が必要です。

Generate manifest

オブジェクトフィルター条件に基づいてマニフェストを自動的に生成する S3 バッチオペレーション `S3DeleteObjectTagging` ジョブを作成する方法は、次の例のとおりです。この条件には、作成日、キー名、サイズ、ストレージクラス、タグが含まれます。

バッチ操作 **`S3DeleteObjectTagging`** ジョブを作成するには

1. 次のコマンドを使用して AWS Identity and Access Management (IAM) ロールを作成してから、IAM ポリシーを作成してアクセス許可を割り当てます。次のロールとポリシーは、オブジェクト タグを削除するための Amazon S3 アクセス許可を付与します。このアクセス許可は、以降のステップでジョブを作成する際に必要になります。

   a. 次のコマンド例を使用して、バッチオペレーションで使用する IAM ロールを作成します。このコマンド例を使用するには、*`S3BatchJobRole`* をロールにつける名前に置き換えます。

```
aws iam create-role \
 --role-name S3BatchJobRole \
 --assume-role-policy-document '{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}'
```

   ロールの Amazon リソースネーム (ARN) を記録します。ジョブの作成時にこの ARN が必要となります。

b. 次のコマンド例を使用して、必要なアクセス許可を持つ IAM ポリシーを作成して、前の手順で作成した IAM ロールにアタッチします。必要なアクセス権限の詳細については、「Amazon S3 バッチ操作に対するアクセス許可の付与」を参照してください。

> **Note**
>
> ディレクトリバケットに対してアクションを実行するバッチオペレーションジョブには、特定のアクセス許可が必要です。詳細については、S3 Express One Zone 向け AWS Identity and Access Management (IAM)」を参照してください。

このサンプルコマンドを使用するには、*user input placeholders* を次のとおり置き換えます。

- *S3BatchJobRole* は、IAM ロール名に置き換えます。この名前が以前に使用した名前と一致することを確認します。
- *DeleteObjectTaggingBatchJobPolicy* は、IAM ポリシーにつける名前に置き換えます。
- *example-s3-destination-bucket* は、タグを付けるオブジェクトがあるバケット名と置き換えます。
- *DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET* は、マニフェストの保存先バケット名と置き換えます。
- *DOC-EXAMPLE-REPORT-BUCKET* は、完了レポートの配信先のバケット名と置き換えます。

```
aws iam put-role-policy \
  --role-name S3BatchJobRole \
  --policy-name DeleteObjectTaggingBatchJobPolicy \
  --policy-document '{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:DeleteObjectTagging",
        "s3:DeleteObjectVersionTagging"
      ],
```

```
      "Resource": "arn:aws:s3:::example-s3-destination-bucket/*"
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutInventoryConfiguration"
      ],
      "Resource": "arn:aws:s3:::example-s3-destination-bucket"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:ListBucket"
      ],
      "Resource":[
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET/*",
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET/*"
      ]
    }
  ]
}'
```

2. 次のコマンド例を使用して、S3DeleteObjectTagging ジョブを作成します。

   この例では、--report セクション内の値によって、生成されるジョブレポートのバケット、プレフィックス、形式、範囲が指定されます。--manifest -generator セクションでは、ジョブの対象となるオブジェクトを含むソースバケットに関する情報、ジョブに対して生成されるマニフェスト出力リストに関する情報、マニフェストに含めるオブジェクトの

範囲を作成日、名前の制約、サイズ、ストレージクラスによって絞り込むためのフィルター条件を指定します。このコマンドでは、ジョブの優先度、IAM ロール、AWS リージョン も指定します。

詳細については、AWS CLI コマンドリファレンスの「create-job」を参照してください。

このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。*IAM-role* を先ほど作成した IAM ロールの ARN に置き換えます。

```
aws s3control create-job \
    --account-id 012345678901 \
    --operation '{
        "S3DeleteObjectTagging": {}
    }' \
    --report '{
        "Bucket":"arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET",
        "Prefix":"reports",
        "Format":"Report_CSV_20180820",
        "Enabled":true,
        "ReportScope":"AllTasks"
    }' \
    --manifest-generator '{
        "S3JobManifestGenerator": {
          "ExpectedBucketOwner": "012345678901",
          "SourceBucket": "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
          "EnableManifestOutput": true,
          "ManifestOutputLocation": {
            "ExpectedManifestBucketOwner": "012345678901",
            "Bucket": "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET",
            "ManifestPrefix": "prefix",
            "ManifestFormat": "S3InventoryReport_CSV_20211130"
          },
          "Filter": {
            "CreatedAfter": "2023-09-01",
            "CreatedBefore": "2023-10-01",
            "KeyNameConstraint": {
              "MatchAnyPrefix": [
                "prefix"
              ],
              "MatchAnySuffix": [
                "suffix"
              ]
            },
```

```
            "ObjectSizeGreaterThanBytes": 100,
            "ObjectSizeLessThanBytes": 200,
            "MatchAnyStorageClass": [
              "STANDARD",
              "STANDARD_IA"
            ]
          }
        }
      }' \
    --priority 2 \
    --role-arn IAM-role \
    --region us-east-1
```

応答として、Amazon S3 はジョブ ID (など 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c) を返します。ジョブの識別、モニタリング、変更にはこのジョブ ID が必要です。

AWS SDK for Java の使用

Specify manifest

既存のマニフェスト ファイルにリストされているオブジェクトに対して動作する S3 バッチオペレーション S3PutObjectTagging ジョブを作成する方法は、次の例のとおりです。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Example

```
package aws.example.s3control;



import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.*;

import java.util.UUID;
import java.util.ArrayList;
```

```
import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateJob {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String iamRoleArn = "IAM Role ARN";
        String reportBucketName = "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET";
        String uuid = UUID.randomUUID().toString();

        ArrayList tagSet = new ArrayList<S3Tag>();
        tagSet.add(new S3Tag().withKey("keyOne").withValue("ValueOne"));


        try {
            JobOperation jobOperation = new JobOperation()
                    .withS3PutObjectTagging(new S3SetObjectTaggingOperation()
                            .withTagSet(tagSet)
                    );

            JobManifest manifest = new JobManifest()
                    .withSpec(new JobManifestSpec()
                            .withFormat("S3BatchOperations_CSV_20180820")
                            .withFields(new String[]{
                                    "Bucket", "Key"
                            }))
                    .withLocation(new JobManifestLocation()
                            .withObjectArn("arn:aws:s3:::my_manifests/manifest.csv")
                            .withETag("60e460c9d1046e73f7dde5043ac3ae85"));
            JobReport jobReport = new JobReport()
                    .withBucket(reportBucketName)
                    .withPrefix("reports")
                    .withFormat("Report_CSV_20180820")
                    .withEnabled(true)
                    .withReportScope("AllTasks");

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.createJob(new CreateJobRequest()
                    .withAccountId(accountId)
                    .withOperation(jobOperation)
                    .withManifest(manifest)
```

```
                    .withReport(jobReport)
                    .withPriority(42)
                    .withRoleArn(iamRoleArn)
                    .withClientRequestToken(uuid)
                    .withDescription("job description")
                    .withConfirmationRequired(false)
            );

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Generate manifest

次の例は、作成日、キー名、サイズなどのオブジェクトフィルター条件に基づいてマニフェストを自動的に生成する S3 バッチオペレーション s3PutObjectCopy ジョブを作成する方法は、次の例のとおりです。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Example

```
        package aws.example.s3control;

    import com.amazonaws.AmazonServiceException;
    import com.amazonaws.SdkClientException;
    import com.amazonaws.auth.profile.ProfileCredentialsProvider;
    import com.amazonaws.services.s3control.AWSS3Control;
    import com.amazonaws.services.s3control.AWSS3ControlClient;
    import com.amazonaws.services.s3control.model.CreateJobRequest;
    import com.amazonaws.services.s3control.model.CreateJobResult;
    import com.amazonaws.services.s3control.model.JobManifestGenerator;
    import com.amazonaws.services.s3control.model.JobManifestGeneratorFilter;
    import com.amazonaws.services.s3control.model.JobOperation;
    import com.amazonaws.services.s3control.model.JobReport;
```

```
    import com.amazonaws.services.s3control.model.KeyNameConstraint;
    import com.amazonaws.services.s3control.model.S3JobManifestGenerator;
    import com.amazonaws.services.s3control.model.S3ManifestOutputLocation;
    import com.amazonaws.services.s3control.model.S3SetObjectTaggingOperation;
    import com.amazonaws.services.s3control.model.S3Tag;

    import java.time.Instant;
    import java.util.Date;
    import java.util.UUID;
    import java.util.ArrayList;

    import static com.amazonaws.regions.Regions.US_WEST_2;

    public class test {
        public static void main(String[] args) {
            String accountId = "012345678901";
            String iamRoleArn = "arn:aws:iam::012345678901:role/ROLE";
            String sourceBucketName = "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET";
            String reportBucketName = "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET";
            String manifestOutputBucketName = "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-
OUTPUT-BUCKET";
            String uuid = UUID.randomUUID().toString();
            long minimumObjectSize = 100L;

            ArrayList<S3Tag> tagSet = new ArrayList<>();
            tagSet.add(new S3Tag().withKey("keyOne").withValue("ValueOne"));

            ArrayList<String> prefixes = new ArrayList<>();
            prefixes.add("s3KeyStartsWith");

            try {
                JobOperation jobOperation = new JobOperation()
                        .withS3PutObjectTagging(new S3SetObjectTaggingOperation()
                                .withTagSet(tagSet)
                        );
                S3ManifestOutputLocation manifestOutputLocation = new
 S3ManifestOutputLocation()
                        .withBucket(manifestOutputBucketName)
                        .withManifestPrefix("manifests")
                        .withExpectedManifestBucketOwner(accountId)
                        .withManifestFormat("S3InventoryReport_CSV_20211130");

                JobManifestGeneratorFilter jobManifestGeneratorFilter = new
 JobManifestGeneratorFilter()
```

```
                        .withEligibleForReplication(true)
                        .withKeyNameConstraint(
                                new KeyNameConstraint()
                                        .withMatchAnyPrefix(prefixes))
                        .withCreatedBefore(Date.from(Instant.now()))
                        .withObjectSizeGreaterThanBytes(minimumObjectSize);

                S3JobManifestGenerator s3JobManifestGenerator = new
S3JobManifestGenerator()
                        .withEnableManifestOutput(true)
                        .withManifestOutputLocation(manifestOutputLocation)
                        .withFilter(jobManifestGeneratorFilter)
                        .withSourceBucket(sourceBucketName);

                JobManifestGenerator jobManifestGenerator = new
JobManifestGenerator()
                        .withS3JobManifestGenerator(s3JobManifestGenerator);

                JobReport jobReport = new JobReport()
                        .withBucket(reportBucketName)
                        .withPrefix("reports")
                        .withFormat("Report_CSV_20180820")
                        .withEnabled(true)
                        .withReportScope("AllTasks");

                AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                        .withCredentials(new ProfileCredentialsProvider())
                        .withRegion(US_WEST_2)
                        .build();

                CreateJobResult createJobResult = s3ControlClient.createJob(new
CreateJobRequest()
                        .withAccountId(accountId)
                        .withOperation(jobOperation)
                        .withManifestGenerator(jobManifestGenerator)
                        .withReport(jobReport)
                        .withPriority(42)
                        .withRoleArn(iamRoleArn)
                        .withClientRequestToken(uuid)
                        .withDescription("job description")
                        .withConfirmationRequired(true)
                );

                System.out.println("Created job " + createJobResult.getJobId());
```

```
            } catch (AmazonServiceException e) {
                // The call was transmitted successfully, but Amazon S3 couldn't
process
                // it and returned an error response.
                e.printStackTrace();
            } catch (SdkClientException e) {
                // Amazon S3 couldn't be contacted for a response, or the client
                // couldn't parse the response from Amazon S3.
                e.printStackTrace();
            }
        }
    }
```

REST API の使用

REST API を使用して、バッチ操作ジョブを作成できます。詳細については、「Amazon Simple Storage Service API リファレンス」の「CreateJob」を参照してください。

ジョブのレスポンス

CreateJob リクエストが成功すると、Amazon S3 はジョブ ID を返します。ジョブ ID は Amazon S3 が自動的に生成する一意の識別子であるため、バッチ操作ジョブを識別してそのステータスを監視できます。

AWS CLI、AWS SDK または REST API を介してジョブを作成する場合、S3 バッチオペレーションを設定してジョブを自動処理を開始できます。ジョブはより高い優先度のジョブを待機せずに、準備が整うと直ちに実行されます。

Amazon S3 コンソールを使用してジョブを作成する際は、バッチオペレーションでジョブの処理を開始する前に、ジョブの詳細を確認し、ジョブを実行できるかを確認する必要があります。ジョブが 30 日以上停止状態のままになると、失敗します。

S3 バッチ操作でサポートされるオペレーション

S3 バッチ操作では、数種類のオペレーションがサポートされています。このセクションのトピックでは、これらの各オペレーションについて説明します。

オブジェクトのコピー

コピーオペレーションは、マニフェストで指定した各オブジェクトをコピーします。オブジェクトを同じ AWS リージョンのバケットにコピーしたり、別のリージョンのバケットにコピーしたりできま

す。S3 バッチ操作では、Amazon S3 のオブジェクトのコピーで使用できるほとんどのオプションがサポートされています。このオプションには、オブジェクトのメタデータの設定、アクセス許可の設定およびオブジェクトのストレージクラスの変更が含まれます。

コピーオペレーションを使用して、既存の暗号化されていないオブジェクトをコピーし、同じバケットに暗号化されたオブジェクトとして書き込むこともできます。詳細については、「Encrypting objects with Amazon S3 Batch Operations」を参照してください。

オブジェクトをコピーするときに、オブジェクトのチェックサムの計算に使用されるチェックサムアルゴリズムを変更できます。オブジェクトの追加のチェックサムが計算されていない場合は、使用する Amazon S3 のチェックサムアルゴリズムを指定して追加することもできます。詳細については、「オブジェクトの整合性をチェックする」を参照してください。

Amazon S3 でのオブジェクトのコピー、および必須パラメータとオプションのパラメータの詳細については、このガイドの オブジェクトのコピー、移動、名前の変更 および Amazon Simple Storage Service API リファレンスの CopyObject を参照してください。

制約と制限

- すべてのソースオブジェクトは 1 つのバケットにある必要があります。
- すべての送信先オブジェクトは 1 つのバケットにある必要があります。
- 送信元バケットへの読み込み許可および送信先バケットへの書き込み許可を保持していることが必要です。
- コピーできるオブジェクトのサイズは 5 GB までです。
- S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive クラスから S3 Standard ストレージクラスにオブジェクトをコピーする場合は、まずこれらのオブジェクトを復元する必要があります。詳細については、「アーカイブされたオブジェクトの復元」を参照してください。
- コピージョブは、コピー先リージョン (オブジェクトをコピーする先のリージョン) で作成する必要があります。
- 条件式による ETag の確認とお客様が用意した暗号化キーを使用したサーバー側の暗号化 (SSE-C) を除く、すべてのコピーオプションがサポートされています。
- バケットがバージョン化されていない場合、同じキー名を使用してオブジェクトを上書きできます。
- オブジェクトは、マニフェストに表示されている順序と同じ順序でコピーされるとは限りません。バージョン管理されたバケットでは、現在のバージョンまたは以前のバージョンの順序を維持することが重要な場合は、最初に以前のバージョンをすべてコピーする必要があります。次に、最初のジョブが完了したら、後続のジョブで現在のバージョンをコピーします。

- 低冗長化ストレージ (RRS) クラスへのオブジェクトのコピーはサポートされていません。

S3 バッチ操作を使用したオブジェクトのコピー

S3 バッチ操作を使用して PUT コピージョブを作成し、同じアカウント内または別の送信先アカウントにオブジェクトをコピーできます。次のセクションでは、別のアカウントにあるマニフェストを保存して使用する方法の例を示します。最初のセクションで示しているように、Amazon S3 インベントリからインベントリレポートをコピー先アカウントに配信して、ジョブの作成時に使用できます。または、2 番目の例で示しているように、コピー元またはコピー先アカウントに保存したカンマ区切り値 (CSV) マニフェストを使用できます。3 番目の例は、コピーオペレーションを使用して、既存のオブジェクトに対する S3 バケットキー暗号化を有効にする方法を示しています。

コピーオペレーションの例

- 送信先アカウントに配信されたインベントリレポートを使用して AWS アカウント 間でオブジェクトをコピーする
- ソースアカウントに保存された CSV マニフェストを使用して AWS アカウント 間でオブジェクトをコピーする
- S3 バッチオペレーションを使用した S3 バケットキーによるオブジェクトの暗号化

送信先アカウントに配信されたインベントリレポートを使用して AWS アカウント 間でオブジェクトをコピーする

Amazon S3 インベントリを使用してインベントリレポートを作成し、そのレポートを使用し、S3 バッチ操作を使用してコピーするオブジェクトのリストを作成できます。ソースアカウントまたは送信先アカウントでの CSV マニフェストの使用について詳しくは、the section called “CSV マニフェストを使用して AWS アカウント 間でオブジェクトをコピーする” を参照してください。

Amazon S3 インベントリは、バケット内のオブジェクトのインベントリを生成します。結果のリストは出力ファイルに公開されます。インベントリ対象となるバケットはソースバケットと呼ばれ、インベントリレポートファイルが保存されているバケットは送信先バケットと呼ばれます。

Amazon S3 インベントリレポートは、別の AWS アカウント に配信されるように設定できます。これにより、送信先アカウントでジョブが作成されたときに S3 バッチ操作がインベントリレポートを読み取ることができます。

Amazon S3 インベントリのコピー元およびコピー先バケットの詳細については、「ソースバケットと保存先バケット」を参照してください。

インベントリを設定する最も簡単な方法は、AWS Management Console を使用することですが、REST API、AWS Command Line Interface (AWS CLI)、または AWS SDK を使用することもできます。

以下のコンソールの手順には、S3 バッチ操作ジョブに対するアクセス許可を設定するための大まかなステップが含まれています。この手順では、オブジェクトを送信元アカウントから送信先アカウントにコピーし、インベントリレポートを送信先アカウントに保存します。

さまざまなアカウントが所有する送信元および送信先バケットの Amazon S3 インベントリを設定する方法

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3) を開きます。
2. インベントリレポートを保存する送信先バケットを選択します。

   インベントリレポートを格納するための送信先マニフェストバケットを決定します。この手順では、送信先アカウントは、送信先マニフェストバケットとオブジェクトのコピー先バケットの両方を所有するアカウントです。
3. ソースバケットのオブジェクトを一覧表示するようにインベントリを設定し、そのリストを送信先マニフェストバケットに発行します。

   ソースバケットのインベントリリストを設定します。これを行うときは、リストを保存する送信先バケットを指定します。ソースバケットのインベントリレポートが送信先バケットに発行されます。この手順では、ソースアカウントは、ソースバケットを所有するアカウントです。

   コンソールを使用してインベントリリストを設定する方法またはインベントリリストファイルを暗号化する方法については、「Amazon S3 インベントリの設定」を参照してください。

   出力形式として [CSV] を選択してください。

   送信先バケットの情報を入力したら、[Buckets in another account (別のアカウントのバケット)] を選択します。次に、送信先マニフェストバケットの名前を入力します。必要に応じて、送信先アカウントのアカウント ID を入力できます。

   インベントリ設定が保存されると、コンソールに次のようなメッセージが表示されます。

   Amazon S3 は送信先バケットにバケットポリシーを作成できませんでした。Amazon S3 が送信先バケットに対してデータの保存ができるよう、送信先バケット所有者に次のバケットポリシーの追加を依頼してください。

コンソールには、送信先バケットに使用できるバケットポリシーが表示されます。

4. コンソールに表示される送信先バケットポリシーをコピーします。
5. 送信先アカウントで、インベントリレポートが格納されている送信先マニフェストバケットにコピーしたバケットポリシーを追加します。
6. S3 バッチ操作信頼ポリシーに基づいて、送信先アカウントにロールを作成します。信頼ポリシーの詳細については、「信頼ポリシー」を参照してください。

   ロールの作成について詳しくは、IAM ユーザーガイドのAWS のサービスに許可を委任するロールの作成を参照してください。

   ロールの名前を入力します (ロールの例では BatchOperationsDestinationRoleCOPY という名前を使用しています)。[S3] サービスを選択してから、信頼ポリシーをロールに適用する [S3 バケットバッチ操作] ユースケースを選択します。

   次に [ポリシーの作成]を選択して、ロールに次のポリシーをアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsDestinationObjectCOPY",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectVersionAcl",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectTagging",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::ObjectDestinationBucket/*",
        "arn:aws:s3:::ObjectSourceBucket/*",
        "arn:aws:s3:::ObjectDestinationManifestBucket/*"
      ]
```

```
    }
  ]
}
```

ロールはポリシーを使用して、送信先バケットのマニフェストを読み取るための batchoperations.s3.amazonaws.com アクセス許可を付与します。また、GET オブジェクト、アクセスコントロールリスト (ACL)、タグ、およびソースオブジェクトバケット内のバージョンに対するアクセス許可も付与されます。そしてそれは、PUT オブジェクト、ACL、タグ、およびバージョンに対するアクセス許可を送信先オブジェクトバケットに付与します。

7. ソースアカウントで、前の手順で作成したロールにソースバケット内のオブジェクト、ACL、タグ、およびバージョンの GET を付与する、ソースバケットのバケットポリシーを作成します。このステップにより、S3 バッチ操作は信頼できるロールを介して送信元バケットからオブジェクトを取得できます。

   以下は、ソースアカウントのバケットポリシーの例です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceObjectCOPY",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::DestinationAccountNumber:role/
BatchOperationsDestinationRoleCOPY"
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::ObjectSourceBucket/*"
    }
  ]
}
```

8. インベントリレポートが使用可能になったら、送信先アカウントで S3 バッチ操作 PUT オブジェクトコピージョブを作成し、送信先マニフェストバケットから在庫レポートを選択します。送信先アカウントで作成したロールの ARN が必要です。

   ジョブの作成に関する全般情報については、「S3 バッチオペレーションジョブの作成」を参照してください。

   コンソールを使用してジョブを作成する方法については、「S3 バッチオペレーションジョブの作成」を参照してください。

ソースアカウントに保存された CSV マニフェストを使用して AWS アカウント 間でオブジェクトをコピーする

S3 バッチ操作ジョブのマニフェストとして異なる AWS アカウント に保存されている CSV ファイルを使用できます。S3 インベントリレポートの使用については、「the section called “インベントリレポートを使用して AWS アカウント 間でオブジェクトをコピーする”」を参照してください。

次の手順では、S3 バッチ操作ジョブを使用して、送信元アカウントに保存されている CSV マニフェストファイルを使用して、送信元アカウントから送信先アカウントにオブジェクトをコピーするときにアクセス許可を設定する方法を示します。

異なる AWS アカウント に保存されている CSV マニフェストを設定するには、次のようにします。

1. S3 バッチ操作信頼ポリシーに基づいて、送信先アカウントにロールを作成します。この手順では、送信先アカウントは、オブジェクトのコピー先のアカウントです。

   信頼ポリシーの詳細については、「信頼ポリシー」を参照してください。

   ロールの作成について詳しくは、IAM ユーザーガイドのAWS のサービスに許可を委任するロールの作成を参照してください。

   コンソールを使用してロールを作成する場合は、ロールの名前を入力します (サンプルロールでは `BatchOperationsDestinationRoleCOPY` という名前が使用されています)。[S3] サービスを選択してから、信頼ポリシーをロールに適用する [S3 バケットバッチ操作] ユースケースを選択します。

   次に [ポリシーの作成]を選択して、ロールに次のポリシーをアタッチします。

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "AllowBatchOperationsDestinationObjectCOPY",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectVersionAcl",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectTagging",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::ObjectDestinationBucket/*",
        "arn:aws:s3:::ObjectSourceBucket/*",
        "arn:aws:s3:::ObjectSourceManifestBucket/*"
      ]
    }
  ]
}
```

このポリシーを使用して、ロールはソースマニフェストバケットのマニフェストを読み取るための batchoperations.s3.amazonaws.com アクセス許可を付与します。ソースオブジェクトバケット内の GET オブジェクト、ACL、タグ、およびバージョンに対するアクセス許可を付与します。また、PUT オブジェクト、ACL、タグ、およびバージョンへのアクセス許可を送信先オブジェクトバケットに付与します。

2. ソースアカウントで、前のステップで作成したロールをソースマニフェストバケットの GET オブジェクトとバージョンに付与するマニフェストを含むバケットのバケットポリシーを作成します。

   このステップでは、S3 バッチ操作が信頼できるロールを使用してマニフェストを読み取ることができます。マニフェストを含むバケットにバケットポリシーを適用します。

   以下は、ソースマニフェストバケットに適用するバケットポリシーの例です。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceManfiestRead",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::DestinationAccountNumber:user/ConsoleUserCreatingJob",
          "arn:aws:iam::DestinationAccountNumber:role/
BatchOperationsDestinationRoleCOPY"
        ]
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::ObjectSourceManifestBucket/*"
    }
  ]
}
```

このポリシーでは、送信先アカウントでジョブを作成しているコンソールユーザーに、同じバケットポリシーを介してソースマニフェストバケットでも同じアクセス許可を許可する権限も付与されます。

3. 送信元アカウントで、前の手順で作成したロールに送信元バケット内のオブジェクト、ACL、タグ、およびバージョンの GET を付与する、送信元バケットのバケットポリシーを作成します。S3 バッチ操作は、信頼できるロールを使用して送信元バケットからオブジェクトを取得できます。

   以下は、ソースオブジェクトを含むバケットのバケットポリシーの例です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceObjectCOPY",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::DestinationAccountNumber:role/
BatchOperationsDestinationRoleCOPY"
      },
      "Action": [
```

```
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::ObjectSourceBucket/*"
    }
  ]
}
```

4. 送信先アカウントで S3 バッチ操作ジョブを作成します。送信先アカウントで作成したロールの Amazon リソースネーム (ARN) が必要です。

   ジョブの作成に関する全般情報については、「S3 バッチオペレーションジョブの作成」を参照してください。

   コンソールを使用してジョブを作成する方法については、「S3 バッチオペレーションジョブの作成」を参照してください。

S3 バッチオペレーションを使用した S3 バケットキーによるオブジェクトの暗号化

このセクションでは、Amazon S3 バッチオペレーションのコピーオペレーションを使用して、既存のオブジェクトに対する S3 バケットキーの暗号化を識別して有効にします。S3 バケットキーについて詳しくは、Amazon S3 バケットキーを使用した SSE-KMS のコストの削減 と 新しいオブジェクトで SSE-KMS の S3 バケットキーを使用するようにバケットを設定する を参照してください。

この例で説明するトピックは次のとおりです。

トピック

前提条件

この手順に従うには、AWS アカウント と、作業ファイルと暗号化された結果を保持するための少なくとも 1 つの S3 バケットが必要です。また、以下のトピックを含む、既存の S3 バッチオペレーションのドキュメントの多くが役立つ場合があります。

- S3 バッチ操作の基本
- S3 バッチオペレーションジョブの作成
- S3 バッチ操作でサポートされるオペレーション
- S3 バッチ操作ジョブの管理

ステップ 1: Amazon S3 インベントリを使用してオブジェクトのリストを取得する

まずはじめに、暗号化するオブジェクトを含む S3 バケットを特定し、その内容のリストを取得します。Amazon S3 インベントリレポートは、この実行に最も便利で手頃な方法です。レポートには、バケット内のオブジェクトのリストと、関連付けられているメタデータが表示されます。ソースバケットはインベントリ対象のバケットを示し、送信先バケットは、インベントリレポートファイルを保存するバケットを示します。Amazon S3 インベントリのコピー元およびコピー先バケットの詳細については、「Amazon S3 インベントリ」を参照してください。

インベントリをセットアップする最も簡単な方法は、AWS Management Console を使用することです。ただし、REST API、AWS Command Line Interface (AWS CLI)、または AWS SDK も使用できます。これらのステップを実行する前に、必ずコンソールにサインインして Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開いてください。アクセス許可拒否エラーが発生した場合は、バケットポリシーを送信先バケットに追加します。詳細については、「S3 インベントリおよび S3 分析に対するアクセス許可の付与」を参照してください。

S3 インベントリを使用してオブジェクトのリストを取得するには、次のようにします。

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. ナビゲーションペインで、[Buckets] (バケット) を選択し、暗号化するオブジェクトを含むバケットを選択します。
3. [Management] (管理) タブの [Inventory configurations] (インベントリ設定) セクションに移動し、[Create inventory configuration] (インベントリ設定の作成) を選択します。
4. 新しいインベントリに名前を付け、送信先 S3 バケットの名前を入力し、オプションで Amazon S3 の送信先プレフィックスを作成して、そのバケット内のオブジェクトを割り当てます。
5. [Output format] (出力形式) として [CSV] を選択します。

6. [追加フィールド - オプション] セクションで、[暗号化] と、関心のあるその他の任意のレポートフィールドを選択します。レポート配信の頻度を [Daily] (毎日) に設定して、最初のレポートが早くバケットに配信されるようにします。
7. [Create] (作成) を選択して設定を保存します。

Amazon S3 では最初のレポートが配信されるまでに最大で 48 時間かかることがあります。最初のレポートが到着したら確認してください。最初のレポートを受け取ったら、次のセクションに進み、S3 インベントリレポートの内容をフィルタリングします。このバケットのインベントリレポートを受信する必要がなくなった場合は、S3 インベントリ設定を削除します。削除しない場合、S3 は毎日または毎週のスケジュールでレポートを配信します。

インベントリ・リストは、すべてのオブジェクトの単一のポイント・イン・タイム・ビューではありません。インベントリリストはバケットアイテムのローリングスナップショットであり、最終的には整合します (たとえば、最近追加されたオブジェクトや削除されたオブジェクトはリストに含まれない可能性があります)。静的オブジェクトや、2 日以上前に作成したオブジェクトセットを操作するときには、S3 インベントリと S3 バッチオペレーションの組み合わせが最適です。より新しいデータを操作するには、ListObjectSv2 (バケットの取得) API オペレーションを使用して、オブジェクトのリストを手動で作成します。必要に応じて、このプロセスを数日間、またはインベントリレポートにすべてのキーが必要なステータスで表示されるまで繰り返します。

ステップ 2: S3 Select を使用してオブジェクトリストをフィルタリングする

S3 インベントリレポートを受信後、レポートの内容をフィルタリングして、S3 バケットキーを使用して暗号化されていないオブジェクトのみをリストできます。バケットのすべてのオブジェクトを S3 バケットキーを使用して暗号化する場合は、このステップを無視できます。ただし、この段階で S3 インベントリレポートをフィルタリングすると、前に暗号化したオブジェクトを再暗号化する時間とコストが削減されます。

以下のステップでは、Amazon S3 Select を使用してフィルタリングする方法を示していますが、Amazon Athena を使用することもできます。使用するツールを決定するには、S3 インベントリレポートの `manifest.json` ファイルを開きます。このファイルには、そのレポートに関連付けられているデータファイルの数がリストされます。数が多い場合は、Amazon Athena を使用してください。Amazon Athena は複数の S3 オブジェクトに対して実行されますが、S3 Select は一度に 1 つのオブジェクトに対して動作するためです。Amazon S3 と Athena を一緒に使用する方法について詳しくは、Amazon Athena で Amazon S3 インベントリをクエリする と、ブログ投稿 Encrypting objects with Amazon S3 Batch Operations の Using Athena を参照してください。

S3 Select を使用して S3 インベントリレポートをフィルタリングするには、次のようにします。

1. インベントリレポートの manifest.json ファイルを開き、JSON の fileSchema セクションを探します。このセクションは、データに対して実行するクエリに情報を提供します。

   次の JSON は、バージョニングが有効なバケットの CSV 形式のインベントリに対する manifest.json ファイルの例です。インベントリレポートの設定方法に応じて、マニフェストの内容は異なる場合があります。

```
  {
    "sourceBucket": "batchoperationsdemo",
    "destinationBucket": "arn:aws:s3:::testbucket",
    "version": "2021-05-22",
    "creationTimestamp": "1558656000000",
    "fileFormat": "CSV",
    "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker,
 BucketKeyStatus",
    "files": [
      {
        "key": "demoinv/batchoperationsdemo/DemoInventory/data/009a40e4-
f053-4c16-8c75-6100f8892202.csv.gz",
        "size": 72691,
        "MD5checksum": "c24c831717a099f0ebe4a9d1c5d3935c"
      }
    ]
  }
```

   バケットでバージョニングが有効になっていない場合、または最新バージョンのレポートを実行することを選択した場合、fileSchema は、Bucket、Key、および BucketKeyStatus です。

   バージョニングが有効になっている場合、インベントリレポートのセットアップ方法に応じて、fileSchema に
Bucket、Key、VersionId、IsLatest、IsDeleteMarker、BucketKeyStatus が含まれる場合があります。このため、クエリを実行するときは、列 1、2、3、および 6 に注意してください。

   S3 バッチオペレーションは、ジョブを実行するための入力として、検索条件のフィールド (BucketKeyStatus) に加えて、バケット、キー、およびバージョン ID を必要とします。バージョン ID フィールドは必要ありませんが、バージョニング対応バケットを操作するときには指

定すると便利です。詳細については、「バージョニングが有効なバケットでのオブジェクトの操作」を参照してください。

2. インベントリレポートのデータファイルを見つけます。manifest.json オブジェクトの files の下に、データファイルがリストされます。
3. S3 コンソールでデータファイルを見つけて選択したら、[Actions] (アクション) を選択し、[Query with S3 Select] (S3 Select を使用したクエリ) を選択します。
4. プリセットの [CSV]、[Comma] (カンマ)、および [GZIP] の各フィールドを選択したままにして、[Next] (次へ) を選択します。
5. 先に進む前にインベントリレポートの形式を確認するには、[Show file preview] (ファイルプレビューの表示) を選択します。
6. 参照する列を [SQL expression] (SQL 式) フィールドに入力し、[Run SQL] (SQL の実行) を選択します。次の式は、S3 バケットキー が設定されていないすべてのオブジェクトの列 1 ~ 3 を返します。

   ```
   select s._1, s._2, s._3 from s3object s where s._6 = 'DISABLED'
   ```

   結果の例は次のとおりです。

   ```
         batchoperationsdemo,0100059%7Ethumb.jpg,lsrtIxksLu0R0ZkYPL.LhgD5caTYn6vu
         batchoperationsdemo,0100074%7Ethumb.jpg,sd2M60g6Fdazoi6D5kNARIE7KzUibmHR
         batchoperationsdemo,0100075%7Ethumb.jpg,TLYESLnl1mXD5c4BwiOIinqFrktddkoL
         batchoperationsdemo,0200147%7Ethumb.jpg,amufzfMi_fEw0Rs99rxR_HrDFlE.l3Y0
         batchoperationsdemo,0301420%7Ethumb.jpg,9qGU2SEscL.C.c_sK89trmXYIwooABSh
         batchoperationsdemo,0401524%7Ethumb.jpg,ORnEWNuB1QhHrrYAGFsZhbyvEYJ3DUor
         batchoperationsdemo,200907200065HQ
   %7Ethumb.jpg,d8LgvIVjbDR5mUVwW6pu9ahTfReyn5V4
         batchoperationsdemo,200907200076HQ
   %7Ethumb.jpg,XUT25d7.gK40u_GmnupdaZg3BVx2jN40
         batchoperationsdemo,201103190002HQ
   %7Ethumb.jpg,z.2sVRh0myqVi0BuIrngWlsRPQdb7qOS
   ```

7. 結果をダウンロードし、CSV 形式で保存し、S3 バッチオペレーションジョブのオブジェクトのリストとして Amazon S3 にアップロードします。
8. マニフェストファイルが複数ある場合は、それらに対して S3 Select を使用したクエリも実行します。結果のサイズに応じて、リストを組み合わせて単一の S3 バッチオペレーションジョブを実行することも、各リストを個別のジョブとして実行することもできます。

   実行するジョブ数を決定するときは、各 S3 バッチオペレーションジョブを実行する料金を考慮してください。

ステップ 3: S3 バッチオペレーションジョブをセットアップして実行する

これで、S3 オブジェクトの CSV リストがフィルタリングされたため、S3 バッチオペレーションジョブを開始して、S3 バケットキーを使用してオブジェクトを暗号化できます。

ジョブは、指定されるオブジェクトのリスト (マニフェスト)、実行されるオペレーション、および指定されたパラメータの総称です。このオブジェクトのセットを暗号化する最も簡単な方法は、PUT コピーオペレーションを使用し、マニフェストにリストされているオブジェクトと同じ送信先プレフィックスを指定することです。これにより、バージョニング非対応のバケット内の既存のオブジェクトが上書きされるか、バージョニングがオンになっている場合は新しい暗号化されたバージョンのオブジェクトが作成されます。

オブジェクトのコピーの一環として、Amazon S3 が SSE-KMS 暗号化および S3 を使用してオブジェクトを暗号化するように指定します。このジョブはオブジェクトをコピーするため、S3 に最初に追加した日時に関係なく、完了時にすべてのオブジェクトについて更新された作成日が表示されます。また、S3 バッチオペレーションジョブの一環として、オブジェクトタグやストレージクラスなどのオブジェクトのセットの他のプロパティを指定します。

サブステップ

- IAM ポリシーをセットアップする
- バッチオペレーションの IAM ロールのセットアップ
- 既存のバケットに対して S3 バケットキーを有効にする
- バッチオペレーションジョブを作成する
- バッチオペレーションジョブを実行する
- 主要事項

IAM ポリシーをセットアップする

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで、[Policies] (ポリシー) を選択し、[Create Policy] (ポリシーの作成) を選択します。
3. [JSON] タブを選択します。[Edit policy] (ポリシーの編集) を選択し、次のコードブロックに表示される IAM ポリシーの例を追加します。

   ポリシーの例を IAM コンソールにコピーしたら、次の項目を置き換えます。

   a. *SOURCE_BUCKET_FOR_COPY* をソースバケットの名前に置き換えます。

b. *DESTINATION_BUCKET_FOR_COPY* を、宛先のバケットの名前に置き換えます。

c. *MANIFEST_KEY* を、マニフェストオブジェクトの名前に置き換えます。

d. *REPORT_BUCKET* を、レポートを保存するバケットの名前に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CopyObjectsToEncrypt",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectTagging",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectVersionAcl",
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::SOURCE_BUCKET_FOR_COPY/*",
        "arn:aws:s3:::DESTINATION_BUCKET_FOR_COPY/*"
      ]
    },
    {
      "Sid": "ReadManifest",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::MANIFEST_KEY"
    },
    {
      "Sid": "WriteReport",
      "Effect": "Allow",
      "Action": [
```

```
            "s3:PutObject"
        ],
        "Resource": "arn:aws:s3:::REPORT_BUCKET/*"
      }
    ]
}
```

4. [次へ: タグ] を選択します。
5. 必要なタグを追加し (オプション)、[Next: Review] (次へ: 確認) を選択します。
6. ポリシー名と、オプションで説明を追加追加し、[Create Policy] (ポリシーの作成) を選択します。
7. [Review policy] (ポリシーの確認) を選択し、[Save changes] (変更の保存) を選択します。
8. S3 バッチオペレーションポリシーが完了したため、コンソールが IAM の [Policies] (ポリシー) ページに戻ります。ポリシー名をフィルタリングし、ポリシー名の左にあるボタンを選択します。次に、[Policy actions] (ポリシーアクション) を選択し、[Attach] (アタッチ) を選択します。

   新しく作成したポリシーを IAM ロールにアタッチするには、アカウントで該当するユーザー、グループ、またはロールを選択し、[Attach policy] (ポリシーのアタッチ) を選択します。これにより、IAM コンソールに戻ります。

バッチオペレーションの IAM ロールのセットアップ

1. IAM コンソールのナビゲーションペインで、[ロール]、[ロールの作成] の順に選択します。
2. [AWS のサービス]、[S3]、および [S3 バッチオペレーション] を選択します。続いて、[Next: Permissions] を選択します。
3. 先ほど作成した IAM ポリシーの名前の入力を開始します。表示されるポリシー名のチェックボックスをオンにし、[Next: Tags] (次: タグ) を選択します。
4. (オプション) この演習では、タグを追加するか、キーと値のフィールドを空白のままにします。[Next: Review] を選択します。
5. ロール名を入力し、デフォルトの説明をそのまま使用するか、独自の説明を追加します。[ロールの作成] を選択します。
6. ジョブを作成するユーザーに、次の例の権限があることを確認します。

   {ACCOUNT-ID} を AWS アカウント ID に、*{IAM_ROLE_NAME}* を後でバッチオペレーションジョブの作成ステップで作成する IAM ロールに適用する予定の名前に置き換えます。詳細については、「Amazon S3 バッチ操作に対するアクセス許可の付与」を参照してください。

```
            {
            "Sid": "AddIamPermissions",
            "Effect": "Allow",
            "Action": [
            "iam:GetRole",
            "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::ACCOUNT-ID:role/IAM_ROLE_NAME"
            }
```

既存のバケットに対して S3 バケットキーを有効にする

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. [バケット] リストで、S3 バケットキーを有効にするバケットを選択します。
3. [プロパティ] を選択します。
4. [Default encryption (デフォルトの暗号化)] で、[Edit (編集)] を選択します。
5. [暗号化タイプ] で、[Amazon S3 マネージドキー (SSE-S3)] または [AWS Key Management Service キー (SSE-KMS] を選択できます。
6. [AWS Key Management Service キー (SSE-KMS)] を選択した場合は、AWS KMS key で、以下のオプションのいずれかを使用して AWS KMS キーを指定できます。
   - 使用可能な KMS キーのリストから選択するには、[AWS KMS キーから選択する] を選びます。使用可能なキーのリストから、バケットと同じリージョンの対称暗号化 KMS キーを選択します。AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がリストに表示されます。
   - KMS キー ARN を入力するには、[AWS KMS キー ARN を入力] を選択し、表示されたフィールドに KMS キー ARN を入力します。
   - AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。
7. [Bucket Key] (バケットキー) の [Enable] (有効化) を選択し、 [Save changes] (変更の保存) を選択します。

バケットレベルで S3 バケットキーがオンになったため、このバケットに対してアップロード、変更、またはコピーされたオブジェクトは、デフォルトでこの暗号化設定を継承します。これには、Amazon S3 バッチオペレーションを使用してコピーされたオブジェクトが含まれます。

バッチオペレーションジョブを作成する

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. ナビゲーションペインで、[Batch Operations] (バッチオペレーション) を選択し、[Create Job] (ジョブの作成) を選択します。
3. オブジェクトを保存する [Region] (リージョン) を選択し、マニフェストタイプとして [CSV] を選択します。
4. パスを入力するか、前に S3 Select (または Athena) の結果から作成した CSV マニフェストファイルに移動します。マニフェストにバージョン ID が含まれている場合は、そのボックスを選択します。[Next (次へ)] を選択します。
5. [Copy] (コピー) オペレーションを選択し、コピー先バケットを選択します。サーバー側の暗号化は無効のままにできます。送信先バケットで S3 バケットキーが有効になっている限り、コピーオペレーションは送信先バケットに S3 バケットキーを適用します。
6. (オプション) 必要に応じて、ストレージクラスおよびその他のパラメータを選択します。このステップで指定するパラメータは、マニフェストにリストされているオブジェクトに対して実行されるすべてのオペレーションに適用されます。[Next] を選択します。
7. サーバー側の暗号化を設定するには、次の手順に従います。

   a. [サーバー側の暗号化] で、次のいずれかを選択します。

      - Amazon S3 にオブジェクトを格納するときに、バケット設定をデフォルトのサーバー側暗号化のままにするには、[暗号化キーを指定しない] を選択します。送信先バケットで S3 バケットキーが有効になっている限り、コピーオペレーションは送信先バケットに S3 バケットキーを適用します。

        > **Note**
        >
        > 指定された宛先のバケットポリシーで、Amazon S3 に保存する前にオブジェクトを暗号化する必要がある場合は、暗号化キーを指定する必要があります。そうしないと、宛先へのオブジェクトのコピーが失敗します。

      - オブジェクトを Amazon S3 に保存する前に暗号化するには、[暗号化キーを指定する] を選択します。

   b. [暗号化設定] で [暗号化キーを指定する]] を選択した場合は、[デフォルトの暗号化に送信先バケット設定を使用する] または [デフォルトの暗号化の送信先バケット設定を上書きする]を選択する必要があります。

c. [デフォルトの暗号化に送信先バケット設定を使用する] を選択した場合は、次の暗号化設定を設定する必要があります。

i. [暗号化タイプ] で、[Amazon S3 マネージドキー (SSE-S3)] または [AWS Key Management Service キー (SSE-KMS)] を選択する必要があります。SSE-S3 は、最強のブロック暗号の 1 つである 256 ビットの 高度暗号化規格 (AES-256) を使用して、各オブジェクトを暗号化します。SSE-KMS を使用すると、キーをより細かく制御できます。詳細については、Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)およびAWS KMS キーによるサーバー側の暗号化 (SSE-KMS) の使用を参照してください。

ii. [AWS Key Management Service キー (SSE-KMS)] を選択した場合は、AWS KMS key で、以下のオプションのいずれかを使用して AWS KMS key を指定できます。

- 使用可能な KMS キーのリストから選択するには、[AWS KMS keys から選択する] を選択し、次に、バケットと同じリージョンの対称暗号化 KMS キーを選択します。AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がリストに表示されます。
- KMS キー ARN を入力するには、[AWS KMS キー ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。
- AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

iii. [バケットキー] で [Enable (有効化)] を選択します。コピーオペレーションでは、送信先バケットに、S3 バケットキーが適用されます。

8. ジョブに説明を付け (またはデフォルトのままにし)、その優先順位レベルを設定し、レポートタイプを選択して、[Path to completion report destination] (完了レポートの送信先へのパス) を指定します。

9. [Permissions] (アクセス権限) セクションで、前に定義したバッチオペレーションの IAM ロールを選択するようにしてください。[Next (次へ)] を選択します。

10. [Review] (確認) で、設定を確認します。変更が必要な場合は、[Previous] (前へ) を選択します。バッチオペレーションの設定を確認したら、[Create job] (ジョブの作成) を選択します。

詳細については、「S3 バッチオペレーションジョブの作成」を参照してください。

バッチオペレーションジョブを実行する

セットアップウィザードによって、Amazon S3 コンソールの S3 バッチオペレーションセクションに自動的に戻ります。S3 がプロセスを開始すると、新しいジョブが [New] (新規) 状態から [Preparing] (準備中) 状態に遷移します。[Preparing] (準備中) 状態の間、S3 はジョブのマニフェストを読み取り、エラーをチェックし、オブジェクトの数を計算します。

1. 進行状況を確認するには、Amazon S3 コンソールの [Refresh] (更新) ボタンを選択します。マニフェストのサイズによっては、読み取りに数分または数時間かかることがあります。
2. S3 がジョブのマニフェストの読み取りを終了すると、ジョブは [Awaiting your confirmation] (確認待ち) 状態に移行します。ジョブ ID の左にあるオプションボタンを選択し、[Run job] (ジョブの実行) を選択します。
3. ジョブの設定を確認し、右下隅の [Run job] (ジョブの実行) を選択します。

   ジョブの実行が開始したら、[refresh] (更新) ボタンを選択してコンソールのダッシュボードビューで、または特定のジョブを選択して、進行状況を確認できます。
4. ジョブが完了したら、[Successful] (成功) と [Failed] (失敗) のオブジェクト数を表示して、すべてが予期したとおりに実行されたことを確認できます。ジョブレポートを有効にした場合は、ジョブレポートで、失敗したオペレーションの正確な原因を確認します。

   これらのステップは、AWS CLI、AWS SDK、または Amazon S3 REST API を使用して実行することもできます。ジョブのステータスと完了レポートの追跡については、ジョブステータスと完了レポートの追跡 を参照してください。

主要事項

S3 バッチオペレーションを使用し、S3 バケットキーを使用してオブジェクトを暗号化するときは、次の問題を考慮してください。

- S3 バッチオペレーションがお客様に代わって実行するデータ転送、リクエスト、その他の料金などのオペレーションに関連する料金に加えて、S3 バッチオペレーションジョブ、オブジェクト、およびリクエストに対して課金されます。詳細については、「Amazon S3 の料金」を参照してください。
- バージョニングされたバケットを使用する場合、実行される各 S3 バッチオペレーションジョブは、オブジェクトの新しい暗号化されたバージョンを作成します。また、S3 バケットキーが設定されていない、前のバージョンも維持されます。古いバージョンを削除するには、ライフサイクル設定の要素 で説明されているように、最新でないバージョンに対して S3 ライフサイクルの有効期限ポリシーをセットアップします。

- コピーオペレーションでは、新しい作成日を持つ新しいオブジェクトが作成されます。これは、アーカイブなどのライフサイクルアクションに影響を与える可能性があります。バケット内のすべてのオブジェクトをコピーすると、新しいすべてのコピーの作成日が同一になるか類似する日付になります。これらのオブジェクトをさらに識別し、さまざまなデータサブセットに対して異なるライフサイクルルールを作成するには、オブジェクトタグの使用を検討してください。

[概要]

このセクションでは、既存のオブジェクトをソートして、既に暗号化されているデータをフィルタリングで除外しました。次に、S3 バッチオペレーションを使用して、S3 バケットキーを有効にしたバケットに既存のデータをコピーすることにより、暗号化されていないオブジェクトに S3 バケットキー機能を適用しました。このプロセスにより、既存のオブジェクトをすべて暗号化するなどのオペレーションを完了しながら、時間とコストを節約できます。

S3 バッチオペレーションについて詳しくは、Amazon S3 オブジェクトでの大規模なバッチ操作の実行 を参照してください。

AWS CLI および AWS SDK for Java を使用したタグによるコピーオペレーションの例については、ラベル付けに使用されるジョブタグを使用したバッチ操作ジョブの作成 を参照してください。

AWS Lambda 関数の呼び出し

AWS Lambda の呼び出し関数は、AWS Lambda 関数を開始し、マニフェストにリストされているオブジェクトに対してカスタムアクションを実行します。このセクションでは、S3 バッチ操作で使用する Lambda 関数とその関数を呼び出すジョブを作成する方法について説明します。S3 バッチ操作ジョブでは、`LambdaInvoke` オペレーションを使用して、マニフェストにリストされているすべてのオブジェクトに対して Lambda 関数を実行します。

Lambda 用の S3 バッチ操作は、AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API で操作できます。Lambda の使用について詳しくは、AWS Lambda 開発者ガイドのAWS Lambda の使用を開始するを参照してください。

以下のセクションでは、S3 バッチ操作で Lambda を使用する方法について説明します。

トピック

- Amazon S3 バッチ操作での Lambda の使用
- S3 バッチ操作で使用する Lambda 関数の作成
- Lambda 関数を呼び出す S3 バッチ操作のジョブの作成

- Lambda のマニフェストでのタスクレベルの情報の指定
- S3 Batch 操作のチュートリアルから学ぶ

Amazon S3 バッチ操作での Lambda の使用

S3 バッチ操作で AWS Lambda を使用するときは、S3 バッチ操作で使用するための Lambda 関数を新しく作成する必要があります。既存の Amazon S3 のイベントベースの関数を S3 バッチ操作で再利用することはできません。イベント関数はメッセージの受信のみ可能です。メッセージを返すことはできません。S3 バッチ操作で使用する Lambda 関数では、メッセージを受け取って返す必要があります。Amazon S3 のイベントで Lambda を使用する方法について詳しくは、AWS Lambda 開発者ガイドのAWS Lambda を Amazon S3 に使用するを参照してください。

Lambda 関数を呼び出す S3 バッチ操作のジョブを作成します。このジョブは、マニフェストにリストされているすべてのオブジェクトに対して同じ Lambda 関数を実行します。マニフェストのオブジェクトの処理時に使用する Lambda 関数のバージョンは指定できます。S3 バッチ操作では、非修飾 Amazon リソースネーム (ARN)、エイリアス、特定のバージョンがサポートされています。詳しくは、AWS Lambda 開発者ガイドのAWS Lambda 関数のバージョンを参照してください。

S3 バッチ操作のジョブにエイリアスまたは $LATEST 修飾子を使用する関数の ARN を指定し、それらのいずれかが指すバージョンを更新すると、S3 バッチ操作は新しいバージョンの Lambda 関数を呼び出します。これは、大規模なジョブで機能の一部を更新する場合に役立ちます。S3 バッチ操作で使用するバージョンを変更したくない場合は、ジョブの作成時に `FunctionARN` パラメータで特定のバージョンを指定します。

ディレクトリバケットでの Lambda と Amazon S3 バッチオペレーションの使用

ディレクトリバケットは Amazon S3 バケットタイプの1 つであり、一貫して 1 桁ミリ秒のレイテンシーに維持する必要があるワークロードまたはパフォーマンス重視のアプリケーション向けに設計されています。詳細については、「ディレクトリバケット」を参照してください。

Amazon S3 バッチオペレーションを使用してディレクトリバケットで動作する Lambda 関数を呼び出すには、独自の要件があります。例えば、更新された JSON スキーマを使用して Lambda リクエストを構築し、ジョブを作成する際に `InvocationSchemaVersion` **`2.0`** を指定する必要があります。この更新されたスキーマでは、`UserArguments` にオプションのキーバリューペアを指定できます。これを使用して、既存の Lambda 関数の特定のパラメータを変更できます。詳細については、「AWS ストレージブログ」の「Automate object processing in Amazon S3 directory buckets with S3 Batch Operations and AWS Lambda」を参照してください。

レスポンスと結果のコード

S3 バッチオペレーションは、それぞれに TaskID が関連付けられている 1 つ以上のキーを使用して Lambda 関数を呼び出します。S3 バッチオペレーションには、Lambda 関数からのキーごとの結果コードが必要です。キーごとの結果コードで返されないタスク ID は、リクエストで送信され、treatMissingKeysAs フィールドの結果コードが付与されます。treatMissingKeysAs はオプションのリクエストフィールドで、デフォルトは TemporaryFailure です。次の表に、treatMissingKeysAs フィールドの他の有効な結果コードと値を示します。

Response Code (レスポンスコード)	説明
Succeeded	タスクは正常に完了しました。ジョブ完了レポートをリクエストした場合は、タスクの結果の文字列がレポートに含まれます。
TemporaryFailure	タスクは一時的に失敗し、ジョブが完了する前に再始動されます。結果の文字列は無視されます。最終的な再処理である場合は、エラーメッセージが最終レポートに含まれます。
PermanentFailure	タスクに固定障害が発生しました。ジョブ完了レポートをリクエストした場合、タスクは Failed としてマークされ、エラーメッセージの文字列が含まれます。失敗したタスクの結果の文字列は無視されます。

S3 バッチ操作で使用する Lambda 関数の作成

このセクションでは、Lambda 関数で使用する必要のある AWS Identity and Access Management (IAM) アクセス許可の例について説明します。また、S3 バッチ操作で使用する Lambda 関数の例も示します。Lambda 関数を作成したことがない場合は、AWS Lambda 開発者ガイドのチュートリアル: Amazon S3 トリガーを使用して AWS Lambda 関数を呼び出すを参照してください。

S3 バッチ操作で使用するための Lambda 関数を作成する必要があります。既存の Amazon S3 のイベントベースの関数を再利用することはできません。これは、S3 バッチ操作で使用する Lambda 関数では、特別なデータフィールドを受け取って返す必要があるためです。

> **Important**
>
> Java で作成した AWS Lambda 関数は、ハンドラーインターフェイスとして RequestHandler または RequestStreamHandler のいずれかを受け付けます。ただし、S3 バッチ操作のリクエストとレスポンスの形式をサポートするため、リクエストとレスポンスのカスタムのシリアル化と逆シリアル化のために AWS Lambda は `RequestStreamHandler` インターフェイスを必要とします。このインターフェイスにより、Lambda は Java の `handleRequest` メソッドに InputStream と OutputStream ストリームを渡すことができます。
>
> S3 バッチ操作で Lambda 関数を使用する場合は、必ず `RequestStreamHandler` インターフェイスを使用してください。`RequestHandler` インターフェイスを使用すると、バッチジョブが失敗し、完了レポートに「Invalid JSON returned in Lambda payload」と表示されます。
>
> 詳しくは、AWS Lambda ユーザーガイドのハンドラーのインターフェイスを参照してください。

IAM アクセス許可の例

S3 バッチ操作で Lambda 関数を使用するために必要な IAM アクセス許可の例を以下に示します。

Example - S3 バッチ操作の信頼ポリシー

以下は、バッチ操作の IAM ロールに使用できる信頼ポリシーの例です。この IAM ロールは、ジョブを作成して、バッチ操作に IAM ロールを引き受けるアクセス許可を付与する場合に指定します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "batchoperations.s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Example - Lambda の IAM ポリシー

以下は、S3 バッチ操作に Lambda 関数を呼び出して入力のマニフェストを読み取るアクセス許可を与える IAM ポリシーの例です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "BatchOperationsLambdaPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject",
                "lambda:InvokeFunction"
            ],
            "Resource": "*"
        }
    ]
}
```

リクエストとレスポンスの例

このセクションでは、Lambda 関数のリクエストとレスポンスの例を示します。

Example リクエスト

以下は、Lambda 関数のリクエストの JSON の例です。

```
{
    "invocationSchemaVersion": "1.0",
    "invocationId": "YXNkbGZqYWRmaiBhc2RmdW9hZHNmZGpmaGFzbGtkaGZza2RmaAo",
    "job": {
        "id": "f3cc4f60-61f6-4a2b-8a21-d07600c373ce"
    },
    "tasks": [
        {
            "taskId": "dGFza2lkZ29lc2hlcmUK",
            "s3Key": "customerImage1.jpg",
            "s3VersionId": "1",
            "s3BucketArn": "arn:aws:s3:us-east-1:0123456788:awsexamplebucket1"
        }
    ]
```

```
}
```

Example レスポンス

以下は、Lambda 関数のレスポンスの JSON の例です。

```
{
  "invocationSchemaVersion": "1.0",
  "treatMissingKeysAs" : "PermanentFailure",
  "invocationId" : "YXNkbGZqYWRmaiBhc2RmdW9hZHNmZGpmaGFzbGtkaGZza2RmaAo",
  "results": [
    {
      "taskId": "dGFza2lkZ29lc2hlcmUK",
      "resultCode": "Succeeded",
      "resultString": "[\"Mary Major", \"John Stiles\"]"
    }
  ]
}
```

S3 バッチ操作の Lambda 関数の例

次の Python Lambda の例では、バージョン管理されたオブジェクトから削除マーカーを削除します。

例に示すように、S3 バッチ操作のキーは URL エンコードされます。Amazon S3 を他の AWS のサービスとともに使用するには、S3 バッチ操作から渡されたキーを URL デコードすることが重要です。

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.
```

```
    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
```

```
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
```

```
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

Lambda 関数を呼び出す S3 バッチ操作のジョブの作成

Lambda 関数を呼び出す S3 バッチ操作のジョブを作成する場合は、以下を指定する必要があります。

- Lambda 関数の ARN (関数のエイリアスまたは特定のバージョン番号を含む)
- 関数を呼び出すアクセス許可を持つ IAM ロール
- アクションパラメータ `LambdaInvokeFunction`

S3 バッチ操作のジョブの作成の詳細については、「S3 バッチオペレーションジョブの作成」および「S3 バッチ操作でサポートされるオペレーション」を参照してください。

次の例では、AWS CLI を使用して Lambda 関数を呼び出す S3 バッチ操作のジョブを作成します。

```
aws s3control create-job
    --account-id <AccountID>
    --operation  '{"LambdaInvoke": { "FunctionArn":
 "arn:aws:lambda:Region:AccountID:function:LambdaFunctionName" } }'
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":
{"ObjectArn":"arn:aws:s3:::ManifestLocation","ETag":"ManifestETag"}}'
    --report
 '{"Bucket":"arn:aws:s3:::awsexamplebucket1","Format":"Report_CSV_20180820","Enabled":true,"Pre
    --priority 2
    --role-arn arn:aws:iam::AccountID:role/BatchOperationsRole
    --region Region
    --description “Lambda Function"
```

Lambda のマニフェストでのタスクレベルの情報の指定

S3 バッチ操作で AWS Lambda 関数を使用するときに、操作対象の各タスク/キーに付随する追加データが必要になる場合があります。たとえば、ソースオブジェクトキーと新しいオブジェクトキーの両方を提供したい場合があります。Lambda 関数は、元のキーを新しい名前で新しい S3 バケットにコピーできます。デフォルトでは、Amazon S3 バッチ操作では、ジョブの入力のマニフェストでコピー先のバケットと元のキーのリストのみを指定できます。以下では、マニフェストにデータを追加して、より複雑な Lambda 関数を実行する方法について説明します。

S3 バッチ操作のマニフェストでキーごとのパラメータを指定して Lambda 関数のコードで使用するには、以下に示すように URL エンコードされた形式の JSON を使用します。`key` フィールドは、Amazon S3 のオブジェクトのキーのように Lambda 関数に渡されます。ただし、以下に示すように、Lambda 関数で他の値や複数のキーを含めるように解釈することができます。

> **Note**
>
> マニフェストの `key` フィールドの最大文字数は 1,024 文字です。

Example - 「Amazon S3 のキー」を JSON の文字列に置き換える前のマニフェスト

URL エンコードされたバージョンを、S3 バッチ操作に渡す必要があります。

```
my-bucket,{"origKey": "object1key", "newKey": "newObject1Key"}
my-bucket,{"origKey": "object2key", "newKey": "newObject2Key"}
my-bucket,{"origKey": "object3key", "newKey": "newObject3Key"}
```

Example - URL エンコードされたマニフェスト

この URL エンコードされたバージョンを、S3 バッチ操作に渡す必要があります。URL エンコードされていないバージョンは機能しません。

```
my-bucket,%7B%22origKey%22%3A%20%22object1key%22%2C%20%22newKey%22%3A
%20%22newObject1Key%22%7D
my-bucket,%7B%22origKey%22%3A%20%22object2key%22%2C%20%22newKey%22%3A
%20%22newObject2Key%22%7D
my-bucket,%7B%22origKey%22%3A%20%22object3key%22%2C%20%22newKey%22%3A
%20%22newObject3Key%22%7D
```

Example - Lambda 関数とジョブのレポートに結果が書き込まれるマニフェストの形式

この URL エンコードされたマニフェストの例には、解析する次の Lambda 関数のパイプ区切りオブジェクトキーが含まれています。

```
my-bucket,object1key%7Clower
my-bucket,object2key%7Cupper
my-bucket,object3key%7Creverse
my-bucket,object4key%7Cdelete
```

この Lambda 関数は、パイプ区切りのタスクをエンコードされた Amazon S3 バッチ操作のマニフェストに解析する方法を示しています。タスクは、指定されたオブジェクトに適用されているリビジョンオペレーションを示します。

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.resource("s3")


def lambda_handler(event, context):
    """
    Applies the specified revision to the specified object.

    :param event: The Amazon S3 batch event that contains the ID of the object to
                  revise and the revision type to apply.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None
```

```
    task = event["tasks"][0]
    task_id = task["taskId"]
    # The revision type is packed with the object key as a pipe-delimited string.
    obj_key, revision = parse.unquote(task["s3Key"], encoding="utf-8").split("|")
    bucket_name = task["s3BucketArn"].split(":")[-1]

    logger.info("Got task: apply revision %s to %s.", revision, obj_key)

    try:
        stanza_obj = s3.Bucket(bucket_name).Object(obj_key)
        stanza = stanza_obj.get()["Body"].read().decode("utf-8")
        if revision == "lower":
            stanza = stanza.lower()
        elif revision == "upper":
            stanza = stanza.upper()
        elif revision == "reverse":
            stanza = stanza[::-1]
        elif revision == "delete":
            pass
        else:
            raise TypeError(f"Can't handle revision type '{revision}'.")

        if revision == "delete":
            stanza_obj.delete()
            result_string = f"Deleted stanza {stanza_obj.key}."
        else:
            stanza_obj.put(Body=bytes(stanza, "utf-8"))
            result_string = (
                f"Applied revision type '{revision}' to " f"stanza {stanza_obj.key}."
            )

        logger.info(result_string)
        result_code = "Succeeded"
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            result_code = "Succeeded"
            result_string = (
                f"Stanza {obj_key} not found, assuming it was deleted "
                f"in an earlier revision."
            )
            logger.info(result_string)
        else:
            result_code = "PermanentFailure"
            result_string = (
```

```
                f"Got exception when applying revision type '{revision}' "
                f"to {obj_key}: {error}."
            )
            logger.exception(result_string)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

S3 Batch 操作のチュートリアルから学ぶ

次のチュートリアルでは、Lambda を使用したいくつかのバッチ操作タスクにおけるエンドツーエンドの一連の手順について説明します。

- チュートリアル: S3 バッチオペレーション、AWS Lambda、および AWS Elemental MediaConvert を使用した動画のバッチトランスコーディング

すべてのオブジェクトタグを置換する

すべてのオブジェクトタグの置換のオペレーションは、マニフェストにリストされているすべてのオブジェクトの Amazon S3 オブジェクトタグを置き換えます。Amazon S3 のオブジェクトのタグは、オブジェクトのメタデータを保存するために使用できる文字列のキーと値のペアです。

すべてのオブジェクトタグの置換のジョブを作成するには、適用するタグのセットを指定します。S3 バッチ操作では、すべてのオブジェクトに同じタグセットが適用されます。指定したタグのセットは、マニフェストのオブジェクトに既に関連付けられているすべてのタグのセットを置き換え

ます。S3 バッチ操作では、既存のタグを保持したままのオブジェクトへのタグの追加はサポートされていません。

マニフェスト内のオブジェクトがバージョニング対応のバケット内にある場合は、タグセットを適用してすべてのオブジェクトのバージョンを指定することができます。これを実行するには、マニフェストですべてのオブジェクトにバージョン ID を指定します。オブジェクトのバージョン ID を含めない場合、S3 バッチ操作はすべてのオブジェクトの最新バージョンにタグセットを適用します。

制約と制限

- バッチ操作ジョブを実行するために指定する AWS Identity and Access Management IAM ロールには、基になる Amazon S3 のすべてのオブジェクトタグの置換のオペレーションを実行するためのアクセス権限が必要です。必要なアクセス権限の詳細については、Amazon Simple Storage Service API リファレンスの「PutObjectTagging」をご参照ください。
- S3 バッチ操作は、Amazon S3 の PutObjectTagging オペレーションを使用して、タグをマニフェストの各オブジェクトに適用します。基になるオペレーションに適用されるすべての制約と制限は、S3 バッチ操作ジョブにも適用されます。

コンソールを使用してジョブを作成する方法の詳細については、S3 バッチ操作ジョブの作成を参照してください。

オブジェクトタグの詳細については、このガイドの タグを使用してストレージを分類する、ならびに Amazon Simple Storage Service API リファレンスのPutObjectTagging、GetObjectTagging、および DeleteObjectTagging を参照してください。

すべてのオブジェクトタグを削除する

すべてのオブジェクトタグの削除のオペレーションは、マニフェストにリストされているオブジェクトに現在関連付けられているすべての Amazon S3 オブジェクトタグセットを削除します。S3 バッチ操作では、他のタグを維持したままオブジェクトからタグを削除することはできません。

マニフェスト内のオブジェクトがバージョニングされたバケットにある場合は、オブジェクトの特定のバージョンからタグセットを削除できます。マニフェストですべてのオブジェクトにバージョン ID を指定してこれを実行します。オブジェクトのバージョン ID を含めない場合、S3 バッチ操作はすべてのオブジェクトの最新バージョンからタグセットを削除します。

バッチ操作マニフェストの詳細については、「マニフェストの指定」をご参照ください。

> **Warning**
>
> このジョブを実行すると、マニフェストにリストされているすべてのオブジェクトのすべてのオブジェクトタグセットが削除されます。

制約と制限

- ジョブを実行するために指定する AWS Identity and Access Management (IAM) ロールには、基になる Amazon S3 のオブジェクトタグの削除のオペレーションを実行するためのアクセス権限が必要です。詳細については、Amazon Simple Storage Service API リファレンスの「DeleteObjectTagging」をご参照ください。
- S3 バッチ操作は、Amazon S3 DeleteObjectTagging オペレーションを使用して、マニフェスト内のすべてのオブジェクトからタグセットを削除します。基になるオペレーションに適用されるすべての制約と制限は、S3 バッチ操作ジョブにも適用されます。

ジョブの作成の詳細については、「S3 バッチオペレーションジョブの作成」をご参照ください。

オブジェクトタグの詳細については、このガイドの「すべてのオブジェクトタグを置換する」、ならびに Amazon Simple Storage Service API リファレンスの「PutObjectTagging」、「GetObjectTagging」、および「DeleteObjectTagging」をご参照ください。

アクセスコントロールリストを置き換える

アクセスコントロールリスト (ACL) の置換のオペレーションは、マニフェストにリストされているすべてのオブジェクトの Amazon S3 アクセスコントロールリスト (ACL) を置き換えます。ACL を使用すると、オブジェクトにアクセスできる人物およびその人物が実行できるアクションを定義できます。

S3 バッチ操作では、ユーザー定義のカスタム ACL と Amazon S3 で提供されている一連のアクセス許可が定義済みの既定の ACL がサポートされています。

マニフェスト内のオブジェクトがバージョニング対応のバケット内にある場合は、ACL を適用してすべてのオブジェクトのバージョンを指定することができます。これを実行するには、マニフェストですべてのオブジェクトにバージョン ID を指定します。オブジェクトのバージョン ID を含めない場合、S3 バッチ操作はオブジェクトの最新バージョンに ACL を適用します。

Amazon S3 の ACL の詳細については、アクセスコントロールリスト (ACL) の概要 を参照してください。

S3 ブロックパブリックアクセス

バケット内のすべてのオブジェクトに対するパブリックアクセスを制限する場合は、S3 バッチ操作ではなく、Amazon S3 のパブリックアクセスブロックを使用することをお勧めします。パブリックアクセスブロックは、素早く効果を発揮する単一の簡単なオペレーションを使用して、バケットごとあるいは全アカウントベースでパブリックアクセスを制限することができます。これは、バケットあるいはアカウント内のすべてのオブジェクトに対するパブリックアクセスを制限することが目的の場合には、最適な選択肢となります。S3 バッチ操作は、マニフェストのすべてのオブジェクトにカスタム ACL を適用する必要がある場合に使用します。S3 ブロックパブリックアクセスについての詳細は、Amazon S3 ストレージへのパブリックアクセスのブロック を参照してください。

S3 オブジェクトの所有権

マニフェスト内のオブジェクトがバケット内にあり、オブジェクト所有権にバケット所有者の強制設定を使用している場合、アクセスコントロールリスト (ACL) を置き換えるオペレーションでは、バケット所有者に完全な制御を許可するオブジェクト ACL のみを指定することができます。このオペレーションはオブジェクト ACL のアクセス許可を、他の人 AWS アカウント またはグループに付与できません。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

制約と制限

- アクセスコントロールリストの置換のジョブを実行するために指定するロールには、基になる Amazon S3 `PutObjectAcl` オペレーションを実行するための許可が必要です。必要なアクセス権限の詳細については、Amazon Simple Storage Service API リファレンスの「PutObjectAcl」をご参照ください。
- S3 バッチ操作は、Amazon S3 の `PutObjectAcl` オペレーションを使用して、指定した ACL をマニフェストのすべてのオブジェクトに適用します。したがって、基になる PutObjectAcl オペレーションに適用されるすべての制約と制限は、S3 バッチ操作のアクセスコントロールリストの置換のジョブにも適用されます。

バッチオペレーションを使ってオブジェクトを復元する

Restore オペレーションは、マニフェストにリストされた、アーカイブされた Amazon S3 オブジェクトの復元リクエストを開始します。次のアーカイブされたオブジェクトは、リアルタイムでアクセスする前に、復元される必要があります。

- S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスにアーカイブされたオブジェクト

- Archive アクセス階層または Deep Archive アクセス階層の S3 Intelligent-Tiering ストレージクラスを介してアーカイブされたオブジェクト

S3 バッチ操作ジョブで S3 オブジェクトの復元の開始のオペレーションを使用すると、マニフェストで指定されたすべてのオブジェクトの復元リクエストが生成されます。

> **Important**
>
> S3 オブジェクトの復元の開始のジョブは、オブジェクトの復元のリクエストのみを開始します。S3 バッチ操作では、各オブジェクトに対するリクエストが開始されると、そのオブジェクトのジョブは完了としてレポートされます。Amazon S3 では、ジョブを更新したり、オブジェクトが復元されたときに通知したりすることはありません。ただし、S3 イベント通知を使用して、Amazon S3 でオブジェクトが利用可能になったときに通知を受けることはできます。詳細については、「Amazon S3 イベント通知」を参照してください。

S3 オブジェクトの復元の開始ジョブを作成するために、次の引数を使用できます。

ExpirationInDays

この引数は、S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive オブジェクトが Amazon S3 で使用できる期間を指定します。S3 Glacier Flexible Retrieval オブジェクトと S3 Glacier Deep Archive オブジェクトをターゲットとするオブジェクトの復元の開始のジョブには、`ExpirationInDays` を 1 以上に設定する必要があります。

> **Important**
>
> S3 Intelligent-Tiering Archive Access および Deep Archive アクセス階層オブジェクトをターゲットとする S3 オブジェクトの復元の開始のオペレーションジョブの作成時には `ExpirationInDays` を設定しないでください。S3 Intelligent-Tiering アーカイブアクセス階層のオブジェクトは復元の有効期限の対象にならないため、`ExpirationInDays` を指定すると復元リクエストが失敗します。

GlacierJobTier

Amazon S3 では、`EXPEDITED`、`STANDARD`、および `BULK` の 3 つの異なる取得階層のいずれかを使用してオブジェクトを復元できます。ただし、S3 バッチ操作機能は、`STANDARD` の取得階

層のみをサポートします。取得階層間の相違点の詳細については、「アーカイブの取り出しオプション」を参照してください。

各レベルの料金の詳細については、[Amazon S3 の料金] ページの「リクエストとデータ取り出し」セクションをご参照ください。

S3 Glacier と S3 Intelligent-Tiering からの復元の違い

S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive ストレージクラスからのアーカイブファイルの復元は、Archive アクセス階層または Deep Archive アクセス階層の S3 Intelligent-Tiering ストレージクラスからのファイルの復元とは異なります。

- S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive から復元すると、オブジェクトの一時コピーが作成されます。Amazon S3 は、`ExpirationInDays` 引数で指定した値が経過した後、このコピーを削除します。この一時コピーが削除された後にそのオブジェクトにアクセスするには、追加の復元リクエストを送信する必要があります。
- アーカイブされた S3 Intelligent-Tiering オブジェクトを復元するときは、`ExpirationInDays` 引数を指定しないでください。S3 Intelligent-Tiering Archive アクセス階層または Deep Archive アクセス階層からオブジェクトを復元すると、オブジェクトは S3 Intelligent-Tiering Frequent アクセス階層に戻ります。オブジェクトは、90 日以上連続でアクセスされないと、Archive Access 階層に自動的に移行します。オブジェクトは、180 日以上連続でアクセスされないと、Deep Archive Access 階層に自動的に移行します。
- バッチ操作ジョブは、S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive ストレージクラスオブジェクト、または S3 Intelligent-Tiering Archive Access および Deep Archive Access ストレージ階層オブジェクトのいずれかで操作できます。バッチオペレーションは、同じジョブ内の両方のタイプのアーカイブオブジェクトに対して操作できません。両方のタイプのオブジェクトを復元するには、別個のバッチ操作ジョブを作成する必要があります。

復元の重複

既に復元中のオブジェクトに対して S3 オブジェクトの復元の開始ジョブを実行すると、S3 バッチ操作は次のように処理します。

次のいずれかの条件が true の場合には、オブジェクトに対するこの復元オペレーションは成功します。

- 進行中の復元リクエストと比較して、このジョブの `ExpirationInDays` 値が同じであり、また、`GlacierJobTier` 値が高速であること。

- 以前の復元リクエストが完了済みであり、オブジェクトが現在利用可能であること。この場合、バッチ操作は、復元されたオブジェクトの有効期限を、進行中の復元リクエストで指定された `ExpirationInDays` 値と一致するように更新します。

次のいずれかの条件が true の場合には、オブジェクトに対するこの復元オペレーションは失敗します。

- 進行中の復元リクエストが未完了であり、かつ、このジョブの復元期間 (`ExpirationInDays` 値によって指定) が進行中の復元リクエストで指定される復元期間とは異なること。
- このジョブの復元階層 (`GlacierJobTier` 値によって指定) が進行中の復元リクエストで指定される復元階層と同じであるか、またはより低速であること。

制限事項

S3 オブジェクトの復元開始ジョブには以下の制限があります。

- アーカイブされたオブジェクトと同じリージョンにジョブを作成する必要があります。
- S3 バッチ操作では、`EXPEDITED` 取得階層はサポートされていません。

オブジェクトの復元の詳細については、「アーカイブされたオブジェクトの復元」をご参照ください。

S3 オブジェクトロックの保持

Object Lock 保持オペレーションでは、ガバナンスモードまたはコンプライアンスモードのいずれかを使用して、オブジェクトの保持日を適用できます。これらのリテンションモードは、さまざまなレベルの保護を適用します。いずれかのリテンションモードを任意のオブジェクトバージョンに適用できます。リーガルホールドなどの保持期間に基づいて、オブジェクトの上書きや削除を防ぎます。Amazon S3 は、オブジェクトのメタデータで指定されている retain until date を保存し、その保持期間が経過するまで指定されたバージョンのオブジェクトを保護します。

S3 バッチ操作とオブジェクトロックを使用すると、多くの Amazon S3 のオブジェクトの保持期間を一括で管理できます。マニフェストでターゲットオブジェクトのリストを指定し、そのマニフェストをジョブの完了のためにバッチ操作に送信します。詳細については、S3 オブジェクトロックの「the section called “保持期間”」を参照してください。

S3 バッチ操作の保持期間の適用ジョブは、完了するか、キャンセルするか、エラー状態になるまで実行されます。S3 バッチ操作と S3 オブジェクトロックによる保持は、1 回のリクエストで多くのオブジェクトの保持期限を追加、変更、削除したい場合に使用することをお勧めします。

バッチ操作は、マニフェストのキーを処理する前に、バケットでオブジェクトロックが有効になっているかどうかを確認します。オペレーションと検証を実行するには、バッチ操作がユーザーの代わりにオブジェクトロックを呼び出せるように、IAM ロールに `s3:GetBucketObjectLockConfiguration` アクセス許可と `s3:PutObjectRetention` アクセス許可が必要です。詳細については、「the section called “オブジェクトロックの考慮事項”」を参照してください。

REST API でこのオペレーションを使用する方法については、Amazon Simple Storage Service API リファレンスの「CreateJob」のオペレーションの `S3PutObjectRetention` を参照してください。

AWS Command Line Interface でのこのオペレーションの使用例については、「the section called “オブジェクトロック保持でバッチ操作を使用する”」を参照してください。AWS SDK for Java の例については、「the section called “オブジェクトロック保持でバッチ操作を使用する”」を参照してください。

制約と制限

- S3 バッチ操作では、バケットレベルでの変更は行われません。
- ジョブを実行するバケットでバージョニングと S3 オブジェクトロックを設定する必要があります。
- マニフェストに登録されているすべてのオブジェクトは同じバケットにあることが必要です。
- マニフェストでオブジェクトのバージョンが明示的に指定されていない限り、オペレーションにはオブジェクトの最新バージョンが使用されます。
- この機能を使用するには、IAM ロールに `s3:PutObjectRetention` アクセス許可が必要です。
- `s3:GetBucketObjectLockConfiguration`S3 バケットでオブジェクトロックが有効になっていることを確認するには、 IAM アクセス許可が必要です。
- オブジェクトの保持期間を延長できるのは、`COMPLIANCE` モードの保持期限が適用されている場合のみであり、短縮することはできません。

S3 オブジェクトロックのリーガルホールド

Object Lock リーガルホールドオペレーションを使用すると、オブジェクトバージョンにリーガルホールドを適用できます。保持期間の設定と同様に、リーガルホールドは、オブジェクトバージョンが上書きまたは削除されるのを防ぎます。ただし、リーガルホールドには関連する保持期間はなく、削除するまで有効です。

S3 バッチ操作とオブジェクトロックを使用すると、一度に多くの Amazon S3 のオブジェクトをリーガルホールドにすることができます。これを行うには、マニフェストに対象のオブジェクトをリストし、そのリストをバッチ操作に渡します。S3 バッチ操作のオブジェクトロックによるリーガルホールドの適用ジョブは、完了するか、キャンセルするか、エラー状態になるまで実行されます。

S3 バッチ操作は、マニフェストのキーを処理する前に、S3 バケットでオブジェクトロックが有効になっているかどうかを確認します。オブジェクトのオペレーションとバケットレベルでの検証を実行するには、S3 バッチ操作がユーザーの代わりに S3 オブジェクトロックを呼び出せるように、IAM ロールに `s3:PutObjectLegalHold` と `s3:GetBucketObjectLockConfiguration` が必要です。

リーガルホールドを解除する S3 バッチ操作のジョブを作成するには、リーガルホールドのステータスに Off を指定するだけで構いません。詳細については、「the section called “オブジェクトロックの考慮事項”」を参照してください。

REST API でこのオペレーションを使用する方法については、Amazon Simple Storage Service API リファレンスの「CreateJob」のオペレーションの「`S3PutObjectLegalHold`」をご参照ください。

このオペレーションの使用例については、「AWS SDK for Java の使用」をご参照ください。

制約と制限

- S3 バッチ操作では、バケットレベルでの変更は行われません。
- マニフェストに登録されているすべてのオブジェクトは同じバケットにあることが必要です。
- ジョブを実行するバケットでバージョニングと S3 オブジェクトロックを設定する必要があります。
- マニフェストでオブジェクトのバージョンが明示的に指定されていない限り、オペレーションにはオブジェクトの最新バージョンが使用されます。
- `s3:PutObjectLegalHold`オブジェクトのリーガルホールドを適用または解除するには、IAM ロールに アクセス許可が必要です。

- `s3:GetBucketObjectLockConfiguration`S3 バケットで S3 オブジェクトロックが有効になっていることを確認するには、 IAM アクセス許可が必要です。

- オブジェクトのコピー
- AWS Lambda 関数の呼び出し
- すべてのオブジェクトタグを置換する
- すべてのオブジェクトタグを削除する
- アクセスコントロールリストを置き換える
- バッチオペレーションを使ってオブジェクトを復元する
- S3 オブジェクトロックの保持
- S3 オブジェクトロックのリーガルホールド
- S3 バッチレプリケーションを使用した既存のオブジェクトのレプリケーション

S3 バッチ操作ジョブの管理

Amazon S3 は、作成した S3 バッチ操作ジョブを管理するための一連の堅牢なツールを用意しています。このセクションでは、AWS Management Console、AWS CLI、AWS SDK、または REST API を使用してジョブを管理および追跡するために使用できるオペレーションについて説明します。

トピック

- S3 バッチオペレーションジョブの管理に Simple Storage Service (Amazon S3) コンソールを使用する
- ジョブのリスト取得
- ジョブの詳細の表示
- ジョブの優先度の割り当て

S3 バッチオペレーションジョブの管理に Simple Storage Service (Amazon S3) コンソールを使用する

コンソールを使用して、S3 バッチオペレーションジョブを管理できます。例えば、以下のことが可能です。

- アクティブなジョブとキュージョブを表示する

- ジョブの優先順位を変更する
- ジョブを確認して実行する
- ジョブのクローンを作成する
- ジョブのキャンセル

S3 コンソールを使用して S3 バッチオペレーションジョブを管理するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[バッチ操作] を選択します。
3. 管理する特定のジョブを選択します。

ジョブのリスト取得

S3 バッチ操作ジョブのリストを取得できます。このリストには、まだ終了していないジョブと 90 日以内に終了したばかりのジョブが含まれます。ジョブリストには、ID、説明、優先度、現在のステータス、および成功したタスクと失敗したタスクの数など、各ジョブに関する情報が含まれています。ジョブのリストをステータスでフィルタリングできます。コンソールからジョブリストを取得するときに、説明または ID でジョブを検索して AWS リージョン 別にフィルタリングすることもできます。

Active ジョブと Complete ジョブのリストを取得する

以下の AWS CLI の例では、`Active` および `Complete` ジョブのリストを取得します。

```
aws s3control list-jobs \
    --region us-west-2 \
    --account-id acct-id \
    --job-statuses '["Active","Complete"]' \
    --max-results 20
```

詳細と例については、「AWS CLI コマンドリファレンス」で「list-jobs」を参照してください。

ジョブの詳細の表示

ジョブを一覧表示することで取得できるよりも多くのジョブの情報が必要な場合、単一のジョブのすべての詳細を表示できます。まだ終了していないジョブや、過去 90 日以内に終了したジョブの詳細

を確認できます。ジョブリストで返される情報に加えて、単一のジョブの詳細には次に示すような他の項目が含まれています。

- オペレーションパラメータ
- マニフェストに関する詳細
- 完了レポートに関する情報 (ジョブの作成時にレポートを設定した場合)
- ジョブの実行に割り当てたユーザーロールの Amazon リソースネーム (ARN)

個々のジョブの詳細を表示することで、そのジョブの設定全体にアクセスできます。ジョブの詳細を表示するには、Amazon S3 コンソールまたは AWS Command Line Interface (AWS CLI) を使用します。

Amazon S3 コンソールで S3 バッチオペレーションジョブの説明を取得する

コンソールを使用してバッチオペレーションジョブの説明を表示するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[バッチ操作] を選択します。
3. 詳細を表示する特定のジョブの ID を選択します。

AWS CLI で S3 バッチオペレーションジョブの説明を取得する

次の例では、AWS CLI を使用して S3 バッチオペレーションジョブの説明を取得します。次のコマンド例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control describe-job \
--region us-west-2 \
--account-id acct-id \
--job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

詳細と例については、「AWS CLI コマンドリファレンス」で「describe-job」を参照してください。

ジョブの優先度の割り当て

各ジョブには、数値の優先度 (任意の正の整数) を割り当てることができます。S3 バッチ操作は、割り当てられた優先度に従ってジョブの優先順位を決定します。優先度が高いジョブ (または、優先パ

ラメータの数値が高いジョブ) ほど先に処理されます。優先度は降順に決定されます。たとえば、優先度の値が 10 のジョブキューは、優先度の値が 1 のジョブキューより先にスケジュールされます。

ジョブの実行中にこのジョブの優先順位を変更できます。また、ジョブの実行中に優先順位が高い新しいジョブを送信すると、優先順位の低いジョブが一時停止して優先順位の高いジョブが実行される可能性があります。

ジョブの優先順位を変更しても、ジョブの処理速度には影響しません。

> **Note**
>
> S3 バッチ操作はベストエフォートベースでジョブの優先度を重視します。優先順位の高いジョブは優先順位の低いジョブより優先されるのが一般的ですが、Amazon S3 はジョブの厳密な順序付けを保証しません。

S3 コンソールの使用

AWS Management Console でジョブの優先度を更新する方法

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[バッチ操作] を選択します。
3. 管理する特定のジョブを選択します。
4. [Actions] (アクション) を選択します。ドロップダウンリストから [Update priority] (更新優先度) を選択します。

AWS CLI の使用

以下の例では、AWS CLI を使用してジョブの優先度を更新します。数値が大きいほど、実行優先度が高くなります。

```
aws s3control update-job-priority \
    --region us-west-2 \
    --account-id acct-id \
    --priority 98 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

AWS SDK for Java の使用

次の例では、AWS SDK for Java を使用して S3 バッチ操作ジョブの優先順位を更新します。

ジョブの優先度の詳細については、「ジョブの優先度の割り当て」を参照してください。

Example

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.UpdateJobPriorityRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateJobPriority {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String jobId = "00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.updateJobPriority(new UpdateJobPriorityRequest()
                    .withAccountId(accountId)
                    .withJobId(jobId)
                    .withPriority(98));

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }
}
```

ジョブステータスと完了レポートの追跡

S3 バッチ操作を使用すると、ジョブステータスの表示と更新、通知と口グの追加、ジョブの失敗の追跡、完了レポートの生成を行うことができます。

トピック

ジョブステータス

ジョブの作成および実行後、ジョブは一連のステータスを進行します。ステータスとその間のあり得る移行を次の表に示します。

ステータス	説明	Transitions
New	ジョブを作成すると、ジョブは New ステータスから始まります。	Amazon S3 がマニフェストオブジェクトの処理を開始すると、ジョブは自動的に Preparing ステータスになります。
Preparing	Amazon S3 はマニフェストオブジェクトと他のジョブパラメータを処理してセットアップし、ジョブを実行します。	Amazon S3 がマニフェストと他のパラメータの処理を終了すると、ジョブは自動的に Ready ステータスになりま

ステータス	説明	Transitions
		す。マニフェストに一覧表示されているオブジェクトにおいて指定されたオペレーションの実行を開始する準備が整いました。 ジョブで実行前に確認が必要な場合は (Amazon S3 コンソールを使用してジョブを作成した場合など)、ジョブは `Preparing` から `Suspended` に移行します。ジョブを実行することが確認されるまで、`Suspended` 状態のままになります。
`Suspended`	ジョブには確認が必要ですが、ジョブを実行したいことをまだ確認していません。Amazon S3 コンソールを使用して作成したジョブのみ確認が必要になります。コンソールを使用して作成したジョブは、`Suspended` 後にすぐ `Preparing` 状態になります。ジョブを実行したいことを確認すると、このジョブは `Ready` になり、`Suspended` 状態に戻ることはありません。	ジョブを実行したいことを確認すると、そのステータスは `Ready` に変更します。

ステータス	説明	Transitions
Ready	Amazon S3 はリクエストしたオブジェクトオペレーションの実行を開始する準備ができました。	Amazon S3 がジョブを開始すると、ジョブは自動的に Active ステータスになります。ジョブが Ready 状態のままになる期間は、優先度の高いジョブをすでに実行中であるかどうか、またそのジョブが完了するまでの時間に応じます。
Active	Amazon S3 はマニフェストで一覧表示されているオブジェクトでリクエストされたオペレーションを実行しています。ジョブが Active の間、Amazon S3 コンソール、あるいは REST API、AWS CLI、または AWS SDK の DescribeJob オペレーションを使用してその進捗状況を監視できます。	ジョブがオブジェクトでオペレーションを実行しなくなると、ジョブは Active 状態ではなくなります。ジョブが成功して完了したか失敗した場合などに、これが自動的に起こります。あるいは、ジョブのキャンセルなどのユーザーアクションの結果としても生じます。ジョブが移行する状態は、移行の理由によって異なります。
Pausing	ジョブは別の状態から Paused に移行しています。	Paused ステージが完了すると、ジョブは自動的に Pausing になります。
Paused	ジョブを現在実行中に優先度の高い別のジョブを送信すると、ジョブは Paused になります。	ジョブの実行が完了、失敗、あるいは停止になることを阻止している優先度の高いジョブの後、Paused ジョブは自動的に Active に戻ります。

ステータス	説明	Transitions
Complete	ジョブは、マニフェストのすべてのオブジェクトでリクエストされたオペレーションの実行を終了しました。このオペレーションは、すべてのオブジェクトで成功または失敗することがあります。完了レポートを生成するようにジョブを設定した場合、このレポートはジョブが Complete になるとすぐに利用可能になります。	Complete は終了状態です。ジョブが Complete に達すると、他の状態に移行することはありません。
Cancelling	このジョブは Cancelled に移行しています。	Cancelled ステージが完了すると、ジョブは自動的に Cancelling になります。
Cancelled	ユーザーはジョブのキャンセルをリクエストし、S3 バッチ操作はジョブのキャンセルに成功しました。このジョブは、Amazon S3 に新しいリクエストを送信しません。	Cancelled は終了状態です。ジョブが Cancelled に達した後、他の状態には移行しません。
Failing	このジョブは Failed に移行しています。	Failed ステージが完了すると、ジョブは自動的に Failing になります。
Failed	このジョブは失敗し、もう実行されていません。ジョブの失敗に関する詳細については、「ジョブの失敗の追跡」を参照してください。	Failed は終了状態です。ジョブが Failed に達した後、他の状態には移行しません。

ジョブステータスの更新

次の AWS CLI および SDK for Java の例は、バッチ操作ジョブのステータスを更新します。S3 コンソールを使用してバッチ操作ジョブを管理する方法の詳細については、S3 バッチオペレーションジョブの管理に Simple Storage Service (Amazon S3) コンソールを使用する を参照してください。

AWS CLI の使用

- たとえば、前の `--no-confirmation-required` の例で、`create-job` パラメータを指定しなかった場合、ジョブのステータスを `Ready` に設定してジョブを確定するまでこのジョブは停止状態になります。その後、Amazon S3 によってジョブが実行可能になります。

```
aws s3control update-job-status \
    --region us-west-2 \
    --account-id 181572960644 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
    --requested-job-status 'Ready'
```

- ジョブステータスを `Cancelled` に設定して、ジョブをキャンセルします。

```
aws s3control update-job-status \
     --region us-west-2 \
     --account-id 181572960644 \
     --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
     --status-update-reason "No longer needed" \
     --requested-job-status Cancelled
```

AWS SDK for Java の使用

次の例では、AWS SDK for Java を使用して S3 バッチ操作ジョブのステータスを更新します。

ジョブのステータスの詳細については、「ジョブステータスと完了レポートの追跡」を参照してください。

Example

```
package aws.example.s3control;
```

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.UpdateJobStatusRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateJobStatus {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String jobId = "00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.updateJobStatus(new UpdateJobStatusRequest()
                    .withAccountId(accountId)
                    .withJobId(jobId)
                    .withRequestedJobStatus("Ready"));

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

通知と口グ記録

完了レポートをリクエストすることに加えて、 を使用してバッチ操作のアクティビティをキャプチャ、確認、および監査することもできますAWS CloudTrail バッチ操作は既存の Amazon S3 API を使用してタスクを実行するため、それらのタスクも直接呼び出した場合と同じイベントを出力します。したがって、Amazon S3 で既に使用しているのと同じ通知、口グ記録、および監査ツールとプ

ロセスを使用して、ジョブの進捗状況とジョブのすべてのタスクを追跡および記録することができます。詳細については、次のセクションの例を参照してください。

> **Note**
>
> Amazon S3 バッチ操作は、ジョブの実行中に CloudTrail で管理イベントとデータイベントの両方を生成します。これらのイベントの量は、各ジョブのマニフェスト内のキーの数に応じてスケールされます。詳細については、CloudTrail の料金のページを参照してください。このページには、アカウントで設定した証跡の数に応じて料金がどのように変わるのかを示す例が記載されています。ニーズに合わせてイベントを設定してログに記録する方法については、AWS CloudTrail ユーザーガイドの最初の証跡の作成を参照してください。

Amazon S3 イベントの詳細については、「Amazon S3 イベント通知」を参照してください。

ジョブの失敗の追跡

指定されたマニフェストを読み取れないなど、S3 バッチ操作が正常に実行できない問題が発生した場合、そのジョブは失敗します。ジョブが失敗すると、1 つ以上の障害コードまたは失敗の理由が生成されます。S3 バッチ操作は、ジョブの詳細をリクエストしてそれらを表示できるように、障害コードと理由をジョブとともに保存します。ジョブの完了レポートをリクエストした場合は、そこに障害コードと理由も表示されます。

ジョブが大量の失敗したオペレーションを実行するのを防ぐために、Amazon S3 はすべてのバッチ操作ジョブにタスク失敗のしきい値を設定します。ジョブが 1000 個以上のタスクを実行すると、Amazon S3 はタスクの失敗率を監視します。いずれかの時点で、失敗率 (実行されたタスクの総数に対する失敗したタスクの数の割合) が 50 パーセントを超えると、そのジョブは失敗します。タスク失敗しきい値を超えたためにジョブが失敗した場合、この失敗の原因を識別できます。たとえば、特定のバケットに存在しないいくつかのオブジェクトを誤ってマニフェストに含めてしまうことがあります。誤りを訂正したら、ジョブを再送信できます。

> **Note**
>
> S3 バッチ操作は非同期で動作するため、タスクはマニフェストにリストされているオブジェクトの順序で実行する必要はありません。つまり、どのオブジェクトのタスクが成功して、どれが失敗したかを判断するためにマニフェストの順序を使用することはできません。代わりに、ジョブの完了レポート (リクエストした場合) を調べるか、AWS CloudTrail イベントログを表示して失敗の原因を特定することができます。

完了レポート

ジョブを作成するときに、完了レポートをリクエストできます。S3 バッチ操作が少なくとも 1 つのタスクを正常に呼び出す限り、タスクの実行が完了した後、失敗したとき、またはキャンセルされたときに、Amazon S3 は完了レポートを生成します。完了レポートに、すべてのタスクを含めるか、失敗したタスクのみを含めるかを設定できます。

完了レポートには、ジョブ設定とステータス、およびオブジェクトキーとバージョン、ステータス、エラーコード、およびエラーの説明を含む各タスクの情報が含まれます。完了レポートは、追加の設定を必要としない統合形式でタスクの結果を表示する簡単な方法です。完了レポートは Amazon S3 マネージドキー (SSE-S3) で暗号化されます。完了レポートの例については、「例: S3 バッチ操作完了レポート」を参照してください。

完了レポートを設定していない場合でも、CloudTrail および Amazon CloudWatch を使用して、ジョブとそのタスクをモニタリングして監査できます。詳細については、以下の セクションを参照してください。

トピック

- 例: を使用して Amazon EventBridge の S3 バッチ操作ジョブを追跡するAWS CloudTrail
- 例: S3 バッチ操作完了レポート

例: を使用して Amazon EventBridge の S3 バッチ操作ジョブを追跡するAWS CloudTrail

Amazon S3 バッチ操作ジョブアクティビティは、 にイベントとして記録されますAWS CloudTrail Amazon EventBridge でカスタムルールを作成し、Amazon Simple Notification Service (Amazon SNS) など、選択したターゲット通知リソースにこれらのイベントを送信できます。

> **Note**
>
> イベントを管理するには、Amazon EventBridge が好ましい方法です。Amazon CloudWatch Events と EventBridge は同じ基盤となるサービスと API ですが、EventBridge はより多くの機能を提供します。CloudWatch または EventBridge のいずれかで行った変更は、各コンソールに表示されます。詳細については、「Amazon EventBridge ユーザーガイドを参照してください。

トラッキングの例

- CloudTrail に記録された S3 バッチ操作イベント
- S3 バッチ操作ジョブイベントを追跡するための EventBridge ルール

CloudTrail に記録された S3 バッチ操作イベント

バッチ操作ジョブが作成されると、CloudTrail に `JobCreated` イベントとして記録されます。ジョブは実行されると、処理中に状態が変わり、他の `JobStatusChanged` イベントが CloudTrail に記録されます。CloudTrail コンソールでこれらのイベントを表示できます。CloudTrail の詳細については、AWS CloudTrail ユーザーガイドを参照してください。

> **Note**
>
> CloudTrail には、S3 バッチ操作ジョブ `status-change` イベントのみが記録されます。

Example CloudTrail によって記録された S3 バッチ操作ジョブ完了イベント

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "accountId": "123456789012",
        "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2020-02-05T18:25:30Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "JobStatusChanged",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "s3.amazonaws.com",
    "userAgent": "s3.amazonaws.com",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "f907577b-bf3d-4c53-b9ed-8a83a118a554",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "123412341234",
    "serviceEventDetails": {
        "jobId": "d6e58ec4-897a-4b6d-975f-10d7f0fb63ce",
        "jobArn": "arn:aws:s3:us-west-2:181572960644:job/
d6e58ec4-897a-4b6d-975f-10d7f0fb63ce",
        "status": "Complete",
        "jobEventId": "b268784cf0a66749f1a05bce259804f5",
```

```
        "failureCodes": [],
        "statusChangeReason": []
    }
}
```

S3 バッチ操作ジョブイベントを追跡するための EventBridge ルール

次の例は、Amazon EventBridge でルールを作成して、AWS CloudTrail によって記録された S3 バッチ操作イベントを任意のターゲットにキャプチャする方法を示しています。

そのためには、「イベントに反応する EventBridge ルールの作成」のすべての手順に従ってルールを作成します。必要に応じて、以下の S3 バッチ操作カスタムイベントパターンポリシーを貼り付け、ターゲットサービスを選択します。

S3 バッチオペレーションカスタムイベントパターンポリシー

```
{
    "source": [
        "aws.s3"
    ],
    "detail-type": [
        "AWS Service Event via CloudTrail"
    ],
    "detail": {
        "eventSource": [
            "s3.amazonaws.com"
        ],
        "eventName": [
            "JobCreated",
            "JobStatusChanged"
        ]
    }
}
```

以下の例は、EventBridge イベントルールから Amazon Simple Queue Service (Amazon SQS) に送信された 2 つのバッチ操作イベントです。バッチ操作ジョブは処理中にさまざまな状態 (`New`、`Preparing`、`Active` など) に移行するため、ジョブごとに複数のメッセージを受信することになります。

Example JobCreated サンプルイベント

```
{
```

```
    "version": "0",
    "id": "51dc8145-541c-5518-2349-56d7dffdf2d8",
    "detail-type": "AWS Service Event via CloudTrail",
    "source": "aws.s3",
    "account": "123456789012",
    "time": "2020-02-27T15:25:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventVersion": "1.05",
        "userIdentity": {
            "accountId": "11112223334444",
            "invokedBy": "s3.amazonaws.com"
        },
        "eventTime": "2020-02-27T15:25:49Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "JobCreated",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "s3.amazonaws.com",
        "userAgent": "s3.amazonaws.com",
        "eventID": "7c38220f-f80b-4239-8b78-2ed867b7d3fa",
        "readOnly": false,
        "eventType": "AwsServiceEvent",
        "serviceEventDetails": {
            "jobId": "e849b567-5232-44be-9a0c-40988f14e80c",
            "jobArn": "arn:aws:s3:us-east-1:181572960644:job/
e849b567-5232-44be-9a0c-40988f14e80c",
            "status": "New",
            "jobEventId": "f177ff24f1f097b69768e327038f30ac",
            "failureCodes": [],
            "statusChangeReason": []
        }
    }
}
```

Example JobStatusChanged ジョブ完了イベント

```
{
  "version": "0",
  "id": "c8791abf-2af8-c754-0435-fd869ce25233",
  "detail-type": "AWS Service Event via CloudTrail",
  "source": "aws.s3",
  "account": "123456789012",
```

```
  "time": "2020-02-27T15:26:42Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "eventVersion": "1.05",
    "userIdentity": {
      "accountId": "1111222233334444",
      "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2020-02-27T15:26:42Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "JobStatusChanged",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "s3.amazonaws.com",
    "userAgent": "s3.amazonaws.com",
    "eventID": "0238c1f7-c2b0-440b-8dbd-1ed5e5833afb",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "serviceEventDetails": {
      "jobId": "e849b567-5232-44be-9a0c-40988f14e80c",
      "jobArn": "arn:aws:s3:us-east-1:181572960644:job/
e849b567-5232-44be-9a0c-40988f14e80c",
      "status": "Complete",
      "jobEventId": "51f5ac17dba408301d56cd1b2c8d1e9e",
      "failureCodes": [],
      "statusChangeReason": []
    }
  }
}
```


例: S3 バッチ操作完了レポート

S3 バッチ操作ジョブを作成するときに、すべてのタスクまたは失敗したタスクについてのみ完了レポートをリクエストできます。少なくとも 1 つのタスクが正常に呼び出されている限り、S3 バッチ操作は、完了、失敗、またはキャンセルされたジョブに関するレポートを生成します。

完了レポートには、オブジェクトキーの名前とバージョン、ステータス、エラーコード、およびエラーの説明など、各タスクに関する追加情報が含まれています。失敗した各タスクのエラーの説明を使用して、アクセス許可の問題など、ジョブの作成中に発生した問題を診断できます。

> **Note**
>
> 完了レポートは常に、Amazon S3 マネージドキー (SSE-S3) で暗号化されます。

Example 最上位マニフェストの結果ファイル

次の例に示すように、最上位の manifest.json ファイルには、成功した各レポートの場所と (ジョブに失敗があった場合は) 失敗したレポートの場所が含まれています。

```
{
    "Format": "Report_CSV_20180820",
    "ReportCreationDate": "2019-04-05T17:48:39.725Z",
    "Results": [
        {
            "TaskExecutionStatus": "succeeded",
            "Bucket": "my-job-reports",
            "MD5Checksum": "83b1c4cbe93fc893f54053697e10fd6e",
            "Key": "job-f8fb9d89-a3aa-461d-bddc-ea6a1b131955/
results/6217b0fab0de85c408b4be96aeaca9b195a7daa5.csv"
        },
        {
            "TaskExecutionStatus": "failed",
            "Bucket": "my-job-reports",
            "MD5Checksum": "22ee037f3515975f7719699e5c416eaa",
            "Key": "job-f8fb9d89-a3aa-461d-bddc-ea6a1b131955/results/
b2ddad417e94331e9f37b44f1faf8c7ed5873f2e.csv"
        }
    ],
    "ReportSchema": "Bucket, Key, VersionId, TaskStatus, ErrorCode, HTTPStatusCode,
 ResultMessage"
}
```

Example 失敗したタスクのレポート

失敗したタスクのレポートには、すべての失敗したタスクに関する以下の情報が含まれています。

- Bucket
- Key
- VersionId

- TaskStatus
- ErrorCode
- HTTPStatusCode
- ResultMessage

次のレポート例は、AWS Lambda 関数がタイムアウトし、失敗が失敗しきい値を超えた場合を示しています。これはその後 PermanentFailure としてマークされました。

```
awsexamplebucket1,image_14975,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:21.155Z 2845ca0d-38d9-4c4b-
abcf-379dc749c452 Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_15897,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:29.610Z 2d0a330b-de9b-425f-
b511-29232fde5fe4 Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_14819,,failed,200,PermanentFailure,"Lambda returned function
 error: {""errorMessage"":""2019-04-05T17:35:22.362Z fcf5efde-74d4-4e6d-b37a-
c7f18827f551 Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_15930,,failed,200,PermanentFailure,"Lambda returned function
 error: {""errorMessage"":""2019-04-05T17:35:29.809Z 3dd5b57c-4a4a-48aa-8a35-
cbf027b7957e Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_17644,,failed,200,PermanentFailure,"Lambda
 returned function error: {""errorMessage"":""2019-04-05T17:35:46.025Z
 10a764e4-2b26-4d8c-9056-1e1072b4723f Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_17398,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:44.661Z 1e306352-4c54-4eba-
aee8-4d02f8c0235c Task timed out after 3.00 seconds""}"
```

Example 成功したタスクのレポート

成功したタスクのレポートには、完了したタスクに関する以下の情報が含まれています。

- Bucket
- Key
- VersionId
- TaskStatus
- ErrorCode
- HTTPStatusCode
- ResultMessage

次の例では、Lambda 関数は Amazon S3 オブジェクトを別のバケットに正常にコピーしています。返された Amazon S3 レスポンスは S3 バッチ操作に戻され、最終的な完了レポートに書き込まれます。

```
awsexamplebucket1,image_17775,,succeeded,200,,"{u'CopySourceVersionId':
 'xVR78haVKlRnurYofbTfYr3ufYbktF8h', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 39, tzinfo=tzlocal()), u'ETag':
 '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata': {'HTTPStatusCode':
 200, 'RetryAttempts': 0, 'HostId': 'nXNaClIMxEJzWNmeMNQV2KpjbaCJLn0OGoXWZpuVOFS/
iQYWxb3QtTvzX9SVfx2lA3oTKLwImKw=', 'RequestId': '3ED5852152014362', 'HTTPHeaders':
 {'content-length': '234', 'x-amz-id-2': 'nXNaClIMxEJzWNmeMNQV2KpjbaCJLn0OGoXWZpuVOFS/
iQYWxb3QtTvzX9SVfx2lA3oTKLwImKw=', 'x-amz-copy-source-version-id':
 'xVR78haVKlRnurYofbTfYr3ufYbktF8h', 'server': 'AmazonS3', 'x-amz-request-id':
 '3ED5852152014362', 'date': 'Fri, 05 Apr 2019 17:35:39 GMT', 'content-type':
 'application/xml'}}}"
awsexamplebucket1,image_17763,,succeeded,200,,"{u'CopySourceVersionId':
 '6HjOUSim4Wj6BTcbxToXW44pSZ.40pwq', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 39, tzinfo=tzlocal()),
 u'ETag': '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata':
 {'HTTPStatusCode': 200, 'RetryAttempts': 0, 'HostId': 'GiCZNYr8LHd/
Thyk6beTRP96IGZk2sYxujLe13TuuLpq6U2RD3we0YoluuIdm1PRvkMwnEW1aFc=', 'RequestId':
 '1BC9F5B1B95D7000', 'HTTPHeaders': {'content-length': '234', 'x-amz-id-2':
 'GiCZNYr8LHd/Thyk6beTRP96IGZk2sYxujLe13TuuLpq6U2RD3we0YoluuIdm1PRvkMwnEW1aFc=', 'x-
amz-copy-source-version-id': '6HjOUSim4Wj6BTcbxToXW44pSZ.40pwq', 'server': 'AmazonS3',
 'x-amz-request-id': '1BC9F5B1B95D7000', 'date': 'Fri, 05 Apr 2019 17:35:39 GMT',
 'content-type': 'application/xml'}}}"
awsexamplebucket1,image_17860,,succeeded,200,,"{u'CopySourceVersionId':
 'm.MDD0g_QsUnYZ8TBzVFrp.TmjN8PJyX', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 40, tzinfo=tzlocal()), u'ETag':
 '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata': {'HTTPStatusCode':
 200, 'RetryAttempts': 0, 'HostId': 'F9ooZOgpE5g9sNgBZxjdiPHqB4+0DNWgj3qbsir
+sKai4fv7rQEcF2fBN1VeeFc2WH45a9ygb2g=', 'RequestId': '8D9CA56A56813DF3', 'HTTPHeaders':
 {'content-length': '234', 'x-amz-id-2': 'F9ooZOgpE5g9sNgBZxjdiPHqB4+0DNWgj3qbsir
+sKai4fv7rQEcF2fBN1VeeFc2WH45a9ygb2g=', 'x-amz-copy-source-version-id':
 'm.MDD0g_QsUnYZ8TBzVFrp.TmjN8PJyX', 'server': 'AmazonS3', 'x-amz-request-id':
 '8D9CA56A56813DF3', 'date': 'Fri, 05 Apr 2019 17:35:40 GMT', 'content-type':
 'application/xml'}}}"
```

タグを使用したアクセスのコントロールとジョブのラベル付け

タグを追加することで、S3 バッチ操作ジョブへのラベル付けとアクセスの制御を実行できます。タグを使用して、バッチ操作ジョブの担当者を識別できます。ジョブタグがあることで、ユーザーによ

るジョブのキャンセル、確認状態にあるジョブの有効化、ジョブの優先度レベルの変更を許可したり制限したりできます。タグをアタッチしてジョブを作成し、後でジョブにタグを追加できます。各タグはキーと値のペアであり、ジョブの作成時に追加することも、後で更新することもできます。

> **Warning**
>
> ジョブタグには機密情報や個人データを含めないでください。

以下のタグ付けの例を考えてみます。経理部門にバッチ操作ジョブを作成するとします。`Department` タグに値 `Finance` を割り当ててジョブが作成される場合に、ユーザーに AWS Identity and Access Management の呼び出しを許可する `CreateJob` (IAM) ポリシーを作成できます。さらに、財務部門のメンバーであるすべてのユーザーにそのポリシーをアタッチできます。

この例を進めて、ユーザーに、必要なタグの付いたジョブの優先度を更新することを許可するポリシーや、それらのタグの付いたジョブをキャンセルすることを許可するポリシーを作成できます。詳細については、「the section called “アクセス許可の制御”」を参照してください。

タグは、新しい S3 バッチ操作ジョブの作成時に追加することも、既存のジョブに追加することもできます。

タグには以下の制限があります。

- タグキーが一意である限り、最大 50 個のタグをジョブに関連付けることができます。
- タグキーには最大 128 個の Unicode 文字、タグ値には最大 256 個の Unicode 文字を使用できます。
- キーと値は大文字と小文字が区別されます。

タグの制限の詳細については、AWS Billing and Cost Management ユーザーガイドのユーザー定義タグの制限を参照してください。

S3 バッチ操作ジョブのタグ付けに関連する API オペレーション

Amazon S3 では、S3 バッチ操作ジョブのタグ付けについて次の API オペレーションがサポートされています。

- GetJobTagging — バッチ操作ジョブに関連付けられたタグセットを返します。
- PutJobTagging — ジョブに関連付けられたタグのセットを置き換えます。この API アクションを使用した S3 バッチ操作ジョブタグの管理には、2 つの異なるシナリオがあります。

- ジョブにタグがない — ジョブに一連のタグを追加できます (ジョブに以前のタグがない)。
- ジョブに既存のタグのセットがある — 既存のタグのセットを変更するには、既存のタグのセット全体を置き換えます。または、GetJobTagging により既存のタグのセットを取得し、そのタグのセットに変更を加えて、この API アクションにより既存のタグのセットを、変更したタグのセットに置き換えます。

> **Note**
>
> タグセットを空にしてこのリクエストを送信すると、S3 バッチ操作 によってオブジェクトの既存のタグセットが削除されます。この方法を使用する場合は、階層 1 リクエスト (PUT) に対して料金が発生します。詳細については、「Amazon S3 の料金」を参照してください。
> バッチ操作ジョブの既存のタグを削除するには、DeleteJobTagging アクションをお勧めします。このアクションでは、料金が発生せずに同じ結果が得られるためです。

- DeleteJobTagging — バッチ操作ジョブに関連付けられたタグセットを削除します。

ラベル付けに使用されるジョブタグを使用したバッチ操作ジョブの作成

タグを追加することで、S3 バッチ操作ジョブへのラベル付けとアクセスの制御を実行できます。タグを使用して、バッチ操作ジョブの担当者を識別できます。タグをアタッチしてジョブを作成し、後でジョブにタグを追加できます。詳細については、「the section called “タグの使用”」を参照してください。

AWS CLI の使用

次の AWS CLI の例では、ジョブタグをジョブのラベルとして使用して S3 バッチ操作 S3PutObjectCopy ジョブを作成します。

1. バッチ操作ジョブで実行するアクションまたは OPERATION を選択してから、TargetResource を選択します。

```
read -d '' OPERATION <<EOF
{
  "S3PutObjectCopy": {
    "TargetResource": "arn:aws:s3:::destination-bucket"
  }
}
```

```
EOF
```

2. ジョブに必要なジョブ TAGS を特定します。この場合、2 つのタグ department および FiscalYear を適用し、値 Marketing および 2020 をそれぞれ使用します。

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
  }
]
EOF
```

3. バッチ操作ジョブの MANIFEST を指定します。

```
read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "EXAMPLE_S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::example-bucket/example_manifest.csv",
    "ETag": "example-5dc7a8bfb90808fc5d546218"
  }
}
EOF
```

4. バッチ操作ジョブの REPORT を設定します。

```
read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::example-report-bucket",
  "Format": "Example_Report_CSV_20180820",
  "Enabled": true,
```

```
  "Prefix": "reports/copy-with-replace-metadata",
  "ReportScope": "AllTasks"
}
EOF
```

5. create-job アクションを実行し、前のステップで設定した入力を使用してバッチ操作ジョブを作成します。

```
aws \
    s3control create-job \
    --account-id 123456789012 \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn arn:aws:iam::123456789012:role/batch-operations-role \
    --tags "${TAGS//$'\n'/}" \
    --client-request-token "$(uuidgen)" \
    --region us-west-2 \
    --description "Copy with Replace Metadata";
```

AWS SDK for Java の使用

Example

次の例では、AWS SDK for Java を使用して、タグ付き S3 バッチ操作ジョブを作成します。

```
public String createJob(final AWSS3ControlClient awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::example-manifest-bucket/
manifests/10_manifest.csv";
    final String manifestObjectVersionId = "example-5dc7a8bfb90808fc5d546218";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new
 JobManifestSpec().withFormat(JobManifestFormat.S3InventoryReport_CSV_20161130);

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
```

```
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::example-report-bucket";
    final String jobReportPrefix = "example-job-reports";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final String lambdaFunctionArn = "arn:aws:lambda:us-
west-2:123456789012:function:example-function";

    final JobOperation jobOperation = new JobOperation()
            .withLambdaInvoke(new
 LambdaInvokeOperation().withFunctionArn(lambdaFunctionArn));

    final S3Tag departmentTag = new
 S3Tag().withKey("department").withValue("Marketing");
    final S3Tag fiscalYearTag = new S3Tag().withKey("FiscalYear").withValue("2020");

    final String roleArn = "arn:aws:iam::123456789012:role/example-batch-operations-
role";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Test lambda job")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withTags(departmentTag, fiscalYearTag)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

S3 バッチ操作ジョブからタグを削除

これらの例を使用して、バッチ操作ジョブからタグを削除できます。

AWS CLI の使用

次の例では、AWS CLI を使用してバッチ操作ジョブからタグを削除します。

```
aws \
    s3control delete-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

バッチ操作ジョブのジョブタグを削除する

Example

次の例では、AWS SDK for Java を使用して S3 バッチ操作ジョブのタグを削除します。

```
public void deleteJobTagging(final AWSS3ControlClient awss3ControlClient,
                             final String jobId) {
    final DeleteJobTaggingRequest deleteJobTaggingRequest = new
 DeleteJobTaggingRequest()
            .withJobId(jobId);

    final DeleteJobTaggingResult deleteJobTaggingResult =
                awss3ControlClient.deleteJobTagging(deleteJobTaggingRequest);
}
```

既存の S3 バッチ操作ジョブにジョブタグを付ける

PutJobTagging を使用して、既存の S3 バッチ操作ジョブにジョブタグを追加できます。詳細については、以下の例を参照してください。

AWS CLI の使用

AWS CLI を使用し、`s3control put-job-tagging` を使用して S3 バッチ操作ジョブにジョブタグを追加する例を次に示します。

> **Note**
>
> タグセットを空にしてこのリクエストを送信すると、S3 バッチ操作 によってオブジェクトの既存のタグセットが削除されます。また、この方法を使用する場合は、階層 1 リクエスト (PUT) に対して料金が発生します。詳細については、「Amazon S3 の料金」を参照してください。
>
> バッチ操作ジョブの既存のタグを削除するには、DeleteJobTagging アクションをお勧めします。このアクションでは、料金が発生せずに同じ結果が得られるためです。

1. ジョブに必要なジョブ TAGS を特定します。この場合、2 つのタグ department および FiscalYear を適用し、値 Marketing および 2020 をそれぞれ使用します。

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
  }
]
EOF
```

2. 必要なパラメータを指定して put-job-tagging アクションを実行します。

```
aws \
    s3control put-job-tagging \
    --account-id 123456789012 \
    --tags "${TAGS//$'\n'/}" \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

AWS SDK for Java の使用

Example

次の例では、AWS SDK for Java を使用して S3 バッチ操作ジョブのタグを配置します。

```
public void putJobTagging(final AWSS3ControlClient awss3ControlClient,
                          final String jobId) {
    final S3Tag departmentTag = new
 S3Tag().withKey("department").withValue("Marketing");
    final S3Tag fiscalYearTag = new S3Tag().withKey("FiscalYear").withValue("2020");

    final PutJobTaggingRequest putJobTaggingRequest = new PutJobTaggingRequest()
            .withJobId(jobId)
            .withTags(departmentTag, fiscalYearTag);

    final PutJobTaggingResult putJobTaggingResult =
 awss3ControlClient.putJobTagging(putJobTaggingRequest);
}
```

S3 バッチ操作ジョブのタグの取得

GetJobTagging を使用して、S3 バッチ操作ジョブのタグを返すことができます。詳細については、以下の例を参照してください。

AWS CLI の使用

次の例では、AWS CLI を使用してバッチ操作ジョブのタグを取得します。

```
aws \
    s3control get-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

AWS SDK for Java の使用

Example

次の例では、AWS SDK for Java を使用してバッチ操作ジョブのタグを取得します。

```
public List<S3Tag> getJobTagging(final AWSS3ControlClient awss3ControlClient,
                                 final String jobId) {
    final GetJobTaggingRequest getJobTaggingRequest = new GetJobTaggingRequest()
            .withJobId(jobId);

    final GetJobTaggingResult getJobTaggingResult =
            awss3ControlClient.getJobTagging(getJobTaggingRequest);
```

```
    final List<S3Tag> tags = getJobTaggingResult.getTags();

    return tags;
}
```

ジョブタグを使用した S3 バッチ操作のアクセス許可の制御

S3 バッチ操作ジョブの管理に役立つように、ジョブタグを追加できます。ジョブタグを使用すると、バッチ操作ジョブへのアクセスをコントロールし、ジョブの作成時にタグが適用されるようにできます。

バッチ操作ジョブごとに最大 50 個のジョブタグを適用できます。これにより、ジョブを編集できるユーザーのセットを制限する非常に細かなポリシーを設定できます。ジョブタグを使用して、ユーザーによるジョブのキャンセル、確認状態にあるジョブの有効化、ジョブの優先度レベルの変更を許可したり制限したりできます。さらに、すべての新しいジョブにタグが適用されるようにし、タグに許可されるキーと値のペアを指定できます。これらのすべての条件を同じ IAM ポリシー言語を使用して表現できます。詳細については、「サービス認証リファレンス」の「Amazon S3 のアクション、リソース、および条件キー」を参照してください。

以下の例では、S3 バッチ操作ジョブタグを使用して、特定の部門(たとえば、財務またはコンプライアンス部門) 内で実行されるジョブのみを作成および編集するアクセス許可をユーザーに付与する方法を示しています。QA や本番稼働など、関連する開発のステージに基づいてジョブを割り当てることもできます。

この例では、AWS Identity and Access Management (IAM) ポリシーで S3 バッチ操作ジョブタグを使用して、部門内で実行されているジョブのみを作成および編集するアクセス権限をユーザーに付与します。QA や本番稼働など、関連する開発のステージに基づいてジョブを割り当てます。

この例では、以下の部門を使用しています。各部門では、バッチ操作をさまざまな方法で使用しています。

- 財務
- コンプライアンス
- ビジネスインテリジェンス
- エンジニアリング

トピック

- ユーザーとリソースへのタグの割り当てによるアクセスのコントロール

- バッチ操作ジョブをステージ別にタグ付けし、ジョブの優先度の制限を強制する

ユーザーとリソースへのタグの割り当てによるアクセスのコントロール

このシナリオでは、管理者は属性ベースのアクセスコントロール (ABAC) を使用しています。ABAC は、ユーザーと AWS リソースの両方にタグを付けることでアクセス許可を定義する IAM 認可戦略です。

ユーザーとジョブには、以下の部門タグのいずれかが割り当てられます。

キー : 値

- `department : Finance`
- `department : Compliance`
- `department : BusinessIntelligence`
- `department : Engineering`

> **Note**
>
> ジョブタグのキーと値は大文字と小文字が区別されます。

ABAC アクセスコントロール戦略を使用して、タグ `department=Finance` をユーザーと関連付けることで、財務部門のユーザーに対して、部門内で S3 バッチ操作ジョブを作成および管理するアクセス許可を付与します。

さらにマネージドポリシーを IAM ユーザーにアタッチして、社内のすべてのユーザーに、それぞれの部門内での S3 バッチ操作ジョブの作成または変更を許可できます。

この例のポリシーには、3 つのポリシーステートメントが含まれています。

- ポリシーの最初のステートメントでは、ユーザーに対して、ジョブ作成リクエストにそれぞれの部門に一致するジョブタグが含まれている場合に、バッチ操作ジョブの作成を許可しています。これは、`"${aws:PrincipalTag/department}"` 構文を使用して表現され、ポリシー評価時にユーザーの部門タグに置き換えられます。リクエスト (`"aws:RequestTag/department"`) の部門タグに指定した値がユーザーの部門と一致すると、条件が満たされます。
- ポリシーの 2 番目のステートメントでは、ユーザーに対して、更新対象のジョブがそのユーザーの部門と一致する場合に、ジョブの優先度の変更、ジョブのステータスの更新を許可しています。

- 3 番目のステートメントでは、ユーザーに対して、(1) 部門タグが保持され、(2) 更新対象のジョブが部門内にある限り、PutJobTagging リクエストによりいつでもバッチ操作ジョブのタグを更新することを許可しています。

```
{
      "Version": "2012-10-17",
      "Statement": [
            {
                  "Effect": "Allow",
                  "Action": "s3:CreateJob",
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:RequestTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
            {
                  "Effect": "Allow",
                  "Action": [
                        "s3:UpdateJobPriority",
                        "s3:UpdateJobStatus"
            ],
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
            {
                  "Effect": "Allow",
                  "Action": "s3:PutJobTagging",
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:RequestTag/department": "${aws:PrincipalTag/
department}",
                              "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
```

```
                }
            }
        }
    ]
}
```

バッチ操作ジョブをステージ別にタグ付けし、ジョブの優先度の制限を強制する

すべての S3 バッチ操作ジョブには数値優先順位があり、Amazon S3 がジョブを実行する順序を決定します。この例では、以下のように、ほとんどのユーザーがジョブに割り当てることができる最大優先度を制限し、高い優先度の範囲を特権のあるユーザーの限定セット用に予約します。

- QA ステージの優先度の範囲 (低): 1 ~ 100
- 本番稼働ステージの優先度の範囲 (高): 1 ~ 300

そのためには、ジョブのステージを表す新しいタグのセットを導入します。

キー : 値

- `stage : QA`
- `stage : Production`

部門内での優先度の低いジョブの作成と更新

このポリシーでは、S3 バッチ操作ジョブの作成と更新に対して、部門ベースの制限に加えて 2 つの新しい制限を導入します。

- ジョブにタグ `stage=QA` が含まれていることを必須とする新しい条件を使用して、ユーザーに、それぞれの部門内でジョブを作成または更新することを許可する。
- ユーザーに、新しい最大優先度 100 までのジョブを作成または更新することを許可する。

```
{
        "Version": "2012-10-17",
        "Statement": [
        {
        "Effect": "Allow",
        "Action": "s3:CreateJob",
        "Resource": "*",
```

```
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/department": "${aws:PrincipalTag/department}",
                "aws:RequestTag/stage": "QA"
            },
            "NumericLessThanEquals": {
                "s3:RequestJobPriority": 100
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:UpdateJobStatus"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:UpdateJobPriority",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}",
                "aws:ResourceTag/stage": "QA"
            },
            "NumericLessThanEquals": {
                "s3:RequestJobPriority": 100
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:PutJobTagging",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/department" : "${aws:PrincipalTag/department}",
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}",
```

```
                "aws:RequestTag/stage": "QA",
                "aws:ResourceTag/stage": "QA"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:GetJobTagging",
        "Resource": "*"
    }
    ]
}
```

部門内での優先度の高いジョブの作成と更新

少数のユーザーのみが QA または本番稼働で優先度の高いジョブを作成できる必要があるとします。この二ーズに応えるには、前のセクションの優先度の低いポリシーを採用した管理ポリシーを作成します。

このポリシーは以下の処理を実行します。

- タグ stage=QA または stage=Production のいずれかを使用して、ユーザーに、部門内でジョブを作成または更新することを許可する。
- ユーザーに、最大優先度 300 までのジョブを作成または更新することを許可する。

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
                "Effect": "Allow",
                "Action": "s3:CreateJob",
                "Resource": "*",
                "Condition": {
                      "ForAnyValue:StringEquals": {
                            "aws:RequestTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                },
                      "StringEquals": {
```

```
                        "aws:RequestTag/department": "${aws:PrincipalTag/
department}"
                },
                      "NumericLessThanEquals": {
                            "s3:RequestJobPriority": 300
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": [
                      "s3:UpdateJobStatus"
            ],
                "Resource": "*",
                "Condition": {
                      "StringEquals": {
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": "s3:UpdateJobPriority",
                "Resource": "*",
                "Condition": {
                      "ForAnyValue:StringEquals": {
                            "aws:ResourceTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                },
                      "StringEquals": {
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                },
                      "NumericLessThanEquals": {
                            "s3:RequestJobPriority": 300
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": "s3:PutJobTagging",
```

```
                "Resource": "*",
                "Condition": {
                      "StringEquals": {
                            "aws:RequestTag/department": "${aws:PrincipalTag/
department}",
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                },
                      "ForAnyValue:StringEquals": {
                            "aws:RequestTag/stage": [
                                  "QA",
                                  "Production"
                    ],
                            "aws:ResourceTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                }
            }
        }
    ]
}
```

S3 バッチ操作を使用した S3 オブジェクトロックの管理

S3 オブジェクトロックを使用すると、オブジェクトバージョンにリーガルホールドを設定することができます。保持期間の設定と同様に、リーガルホールドは、オブジェクトバージョンが上書きまたは削除されるのを防ぎます。ただし、リーガルホールドには関連する保持期間はなく、削除するまで有効です。詳細については、「S3 オブジェクトロックのリーガルホールド」を参照してください。

S3 バッチ操作を Object Lock とともに使用して、多くの Amazon S3 オブジェクトにリーガルホールドを一度に追加する方法については、以下のセクションを参照してください。

トピック

- S3 バッチ操作を使用した S3 オブジェクトロックの有効化
- バッチ操作を使用したオブジェクトロック保持の設定
- S3 オブジェクトロック保持コンプライアンスモードでの S3 バッチ操作の使用
- S3 オブジェクトロック保持ガバナンスモードで S3 バッチ操作を使用する
- S3 バッチ操作を使用した S3 オブジェクトロックの法的保留の無効化

S3 バッチ操作を使用した S3 オブジェクトロックの有効化

S3 バッチ操作を S3 オブジェクトロックとともに使用すると、一度に多くの Amazon S3 オブジェクトの保持を管理したり、法的保留を有効にしたりできます。マニフェストでターゲットオブジェクトのリストを指定し、そのマニフェストをジョブの完了のためにバッチ操作に送信します。詳細については、「the section called “オブジェクトロックの保持”」および「the section called “オブジェクトロックのリーガルホールド”」を参照してください。

以下の例では、S3 バッチ操作アクセス許可を持つ IAM ロールを作成し、そのロールのアクセス許可を更新してオブジェクトロックを有効化するジョブを作成する方法を示します。これらの例では、変数値をニーズに合った値に置き換えます。S3 バッチ操作ジョブのオブジェクトを識別する CSV マニフェストも必要です。詳細については、「the section called “マニフェストの指定”」を参照してください。

AWS CLI の使用

1. IAM ロールを作成し、実行する S3 バッチ操作のアクセス許可を割り当てます。

   このステップは、すべての S3 バッチ操作ジョブで必要です。

```
export AWS_PROFILE='aws-user'

read -d '' bops_trust_policy <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "batchoperations.s3.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
aws iam create-role --role-name bops-objectlock --assume-role-policy-document
 "${bops_trust_policy}"
```

2. S3 オブジェクトロックを使用して S3 バッチ操作を実行するように設定します。

このステップでは、ロールに以下のことを許可します。

a. バッチ操作を実行するターゲットオブジェクトを含む S3 バケットでオブジェクトロックを実行します。

b. マニフェスト CSV ファイルとオブジェクトがある S3 バケットを読み取る。

c. S3 バッチ操作ジョブの結果をレポートバケットに書き込みます。

```
read -d '' bops_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{ReportBucket}}/*"
            ]
        }
    ]
}
```

```
EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name object-lock-
permissions --policy-document "${bops_permissions}"
```

AWS SDK for Java の使用

次の例は、S3 バッチ操作許可を持つ IAM ロールを作成し、AWS SDK for Java を使用してオブジェクトロックを有効にするジョブを作成するためにロールの許可を更新する方法を示しています。コードで、変数値をニーズに合った値に置き換えます。S3 バッチ操作ジョブのオブジェクトを識別する CSV マニフェストも必要です。詳細については、「the section called “マニフェストの指定”」を参照してください。

以下のステップを実行します。

1. IAM ロールを作成し、実行する S3 バッチ操作のアクセス許可を割り当てます。このステップは、すべての S3 バッチ操作ジョブで必要です。
2. S3 オブジェクトロックを使用して S3 バッチ操作を実行するように設定します。

   ロールに以下のことを許可します。

   1. バッチ操作を実行するターゲットオブジェクトを含む S3 バケットでオブジェクトロックを実行します。
   2. マニフェスト CSV ファイルとオブジェクトがある S3 バケットを読み取る。
   3. S3 バッチ操作ジョブの結果をレポートバケットに書き込みます。

```
public void createObjectLockRole() {
    final String roleName = "bops-object-lock";

    final String trustPolicy = "{" +
            "  \"Version\": \"2012-10-17\", " +
            "  \"Statement\": [ " +
            "    { " +
            "      \"Effect\": \"Allow\", " +
            "      \"Principal\": { " +
            "        \"Service\": [" +
            "          \"batchoperations.s3.amazonaws.com\"" +
            "        ]" +
            "      }, " +
```

```
            "      \"Action\": \"sts:AssumeRole\" " +
            "    } " +
            "  ]" +
            "}";

    final String bopsPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": \"s3:GetBucketObjectLockConfiguration\"," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket\"" +
            "            ]" +
            "        }," +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:GetObject\"," +
            "                \"s3:GetObjectVersion\"," +
            "                \"s3:GetBucketLocation\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket/*\"" +
            "            ]" +
            "        }," +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObject\"," +
            "                \"s3:GetBucketLocation\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ReportBucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final CreateRoleRequest createRoleRequest = new CreateRoleRequest()
            .withAssumeRolePolicyDocument(bopsPermissions)
```

```
            .withRoleName(roleName);

    final CreateRoleResult createRoleResult = iam.createRole(createRoleRequest);

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(bopsPermissions)
            .withPolicyName("bops-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

バッチ操作を使用したオブジェクトロック保持の設定

以下の例では、ルールでマニフェストバケット内のオブジェクトの S3 オブジェクトロック保持を設定できるようにします。

`s3:PutObjectRetention` アクセス許可を含めるようにロールを更新すると、バケット内のオブジェクトにオブジェクトロック保持を適用できます。

AWS CLI の使用

```
export AWS_PROFILE='aws-user'

read -d '' retention_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}/*"
            ]
        }
    ]
}
EOF
```

```
aws iam put-role-policy --role-name bops-objectlock --policy-name retention-permissions
 --policy-document "${retention_permissions}"
```

AWS SDK for Java の使用

```
public void allowPutObjectRetention() {
    final String roleName = "bops-object-lock";

    final String retentionPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObjectRetention\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(retentionPermissions)
            .withPolicyName("retention-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

S3 オブジェクトロック保持コンプライアンスモードでの S3 バッチ操作の使用

以下の例は、信頼ポリシーを作成し、オブジェクトに S3 バッチ操作と S3 オブジェクトロックの設定アクセス許可を設定する前の例に基づいています。この例では、保持モードを COMPLIANCE に、retain until date を 2025 年 1 月 1 日に設定しています。また、マニフェストバケット内のオブジェクトをターゲットとし指定したレポートバケットに結果を書き込むジョブを作成します。

AWS CLI の使用

Example 複数のオブジェクト間でのメンションのコンプライアンスの設定

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-01T00:00:00",
      "Mode":"COMPLIANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/compliance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/compliance-objects-bops",
  "ReportScope": "AllTasks"
}
```

```
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Set compliance retain-until to 1 Jul 2030";
```

Example **COMPLIANCE** モードの **retain until date** を 2025 年 1 月 15 日に延長する

以下の例では、COMPLIANCE モードの retain until date を 2025 年 1 月 15 日に延長します。

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-15T00:00:00",
      "Mode":"COMPLIANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
```

```
    "ObjectArn": "arn:aws:s3:::ManifestBucket/compliance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/compliance-objects-bops",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Extend compliance retention to 15 Jan 2025";
```

AWS SDK for Java の使用

Example 保持モードを「コンプライアンス」に設定して、保持期間を 2025 年 1 月 1 日に設定します。

```
public String createComplianceRetentionJob(final AWSS3ControlClient awss3ControlClient)
 throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/compliance-objects-
manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);
```

```
    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/compliance-objects-bops";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date janFirst = format.parse("01/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.COMPLIANCE)
                            .withRetainUntilDate(janFirst)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Set compliance retain-until to 1 Jan 2025")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);
```

```
    return result.getJobId();
}
```

Example **COMPLIANCE** モードの **retain until date** の延長

以下の例では、COMPLIANCE モードの `retain until date` を 2025 年 1 月 15 日に延長します。

```
public String createExtendComplianceRetentionJob(final AWSS3ControlClient
 awss3ControlClient) throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/compliance-objects-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/compliance-objects-bops";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan15th = format.parse("15/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.COMPLIANCE)
                            .withRetainUntilDate(jan15th)));
```

```
    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Extend compliance retention to 15 Jan 2025")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

S3 オブジェクトロック保持ガバナンスモードで S3 バッチ操作を使用する

次の例は、信頼ポリシーを作成し、S3 バッチ操作と S3 オブジェクトロックの設定アクセス許可を設定する前の例に基づいています。これは、2025 年 1 月 30 日の `retain until date` で S3 オブジェクトロック保持ガバナンスを複数のオブジェクトに適用する方法を示しています。また、マニフェストバケットを使用しレポートバケットに結果を書き込むバッチ操作ジョブを作成します。

AWS CLI の使用

Example 2025 年 1 月 30 日まで保持する複数のオブジェクトに S3 オブジェクトロック保持ガバナンスを適用する

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-30T00:00:00",
      "Mode":"GOVERNANCE"
```

```
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/governance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucketT",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/governance-objects",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Put governance retention";
```

Example 複数のオブジェクト間での保持ガバナンスのバイパス

次の例は、信頼ポリシーを作成し、S3 バッチ操作と S3 オブジェクトロックの設定アクセス許可を設定する前の例に基づいています。また、複数のオブジェクト間で保持ガバナンスを省略する方法を示し、マニフェストバケットを使用しレポートバケットに結果を書き込むバッチ操作ジョブを作成します。

```
export AWS_PROFILE='aws-user'

read -d '' bypass_governance_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:BypassGovernanceRetention"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name bypass-governance-
permissions --policy-document "${bypass_governance_permissions}"

export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "BypassGovernanceRetention": true,
    "Retention": {
    }
  }
}
EOF
```

```
read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/governance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::REPORT_BUCKET",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/bops-governance",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Remove governance retention";
```

AWS SDK for Java の使用

次の例は、信頼ポリシーを作成し、S3 バッチ操作と S3 オブジェクトロックの設定アクセス許可を設定する前の例に基づいています。`retain until date` を 2025 年 1 月 30 日に設定して、複数

のオブジェクトにわたって S3 オブジェクトロック保持ガバナンスを適用する方法を示しています。また、マニフェストバケットを使用しレポートバケットに結果を書き込むバッチ操作ジョブを作成します。

Example 2025 年 1 月 30 日まで保持する複数のオブジェクトに S3 オブジェクトロック保持ガバナンスを適用する

```
public String createGovernanceRetentionJob(final AWSS3ControlClient awss3ControlClient)
 throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/governance-objects-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/governance-objects";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan30th = format.parse("30/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.GOVERNANCE)
```

```
                            .withRetainUntilDate(jan30th)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Put governance retention")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

Example 複数のオブジェクト間での保持ガバナンスのバイパス

次の例は、信頼ポリシーを作成し、S3 バッチ操作と S3 オブジェクトロックの設定アクセス許可を設定する前の例に基づいています。また、複数のオブジェクト間で保持ガバナンスを省略する方法を示し、マニフェストバケットを使用しレポートバケットに結果を書き込むバッチ操作ジョブを作成します。

```
public void allowBypassGovernance() {
    final String roleName = "bops-object-lock";

    final String bypassGovernancePermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:BypassGovernanceRetention\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket/*\"" +
            "            ]" +
            "        }" +
```

```
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(bypassGovernancePermissions)
            .withPolicyName("bypass-governance-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
public String createRemoveGovernanceRetentionJob(final AWSS3ControlClient
 awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/governance-objects-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/bops-governance";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final JobOperation jobOperation = new JobOperation()
```

```
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Remove governance retention")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

S3 バッチ操作を使用した S3 オブジェクトロックの法的保留の無効化

以下の例は、信頼ポリシーを作成し、S3 バッチ操作と S3 オブジェクトロックの設定アクセス許可を設定する前の例に基づいています。バッチ操作を使用して、オブジェクトのオブジェクトロックの法的保留を無効にする方法を示します。

この例ではまず、s3:PutObjectLegalHold アクセス許可を付与するようにロールを更新したうえで、マニフェストで指定されたオブジェクトから法的保留をオフにし (削除)、その結果をレポートするバッチ操作ジョブを作成します。

AWS CLI の使用

Example ロールを更新して **s3:PutObjectLegalHold** にアクセス許可を付与する

```
export AWS_PROFILE='aws-user'

read -d '' legal_hold_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectLegalHold"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        }
    ]

EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name legal-hold-
permissions --policy-document "${legal_hold_permissions}"
```

Example リーガルホールドをオフにする

以下の例では、リーガルホールドをオフにします。

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectLegalHold": {
    "LegalHold": {
      "Status":"OFF"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
```

```
    "ObjectArn": "arn:aws:s3:::ManifestBucket/legalhold-object-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/legalhold-objects-bops",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Turn off legal hold";
```

AWS SDK for Java の使用

Example ロールを更新して **s3:PutObjectLegalHold** にアクセス許可を付与する

```
public void allowPutObjectLegalHold() {
    final String roleName = "bops-object-lock";

    final String legalHoldPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObjectLegalHold\"" +
            "            ]," +
            "            \"Resource\": [" +
```

```
            "                \"arn:aws:s3:::ManifestBucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(legalHoldPermissions)
            .withPolicyName("legal-hold-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

Example リーガルホールドをオフにする

リーガルホールドをオフにする場合は、以下の例を使用します。

```
public String createLegalHoldOffJob(final AWSS3ControlClient awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/legalhold-object-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/legalhold-objects-bops";

    final JobReport jobReport = new JobReport()
```

```
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectLegalHold(new S3SetObjectLegalHoldOperation()
                    .withLegalHold(new S3ObjectLockLegalHold()
                            .withStatus(S3ObjectLockLegalHoldStatus.OFF)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Turn off legal hold")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

S3 バッチ操作のチュートリアル

次のチュートリアルでは、いくつかのバッチ操作タスクにおけるエンドツーエンドの一連の手順について説明します。

- チュートリアル: S3 バッチオペレーション、AWS Lambda、および AWS Elemental MediaConvert を使用した動画のバッチトランスコーディング

Amazon S3 のモニタリング

モニタリングは、Amazon S3 および AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできるように、AWS ソリューションのすべての部分からモニタリングデータを収集することをお勧めします。Simple Storage Service (Amazon S3) のモニタリングを開始する前に、以下の質問に対する回答を反映したモニタリング計画を作成する必要があります。

- どのような目的でモニタリングしますか?
- どのリソースをモニタリングしますか?
- どのくらいの頻度でこれらのリソースをモニタリングしますか?
- どのモニタリングツールを利用しますか?
- 誰がモニタリングタスクを実行しますか?
- 問題が発生したときに誰が通知を受け取りますか?

Amazon S3 でのログ記録とモニタリングの詳細については、以下のトピックを参照してください。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

トピック

- モニタリングツール
- Amazon S3 のログ記録オプション
- AWS CloudTrail を使用した Amazon S3 API コールのログ記録
- サーバーアクセスログによるリクエストのログ記録
- Amazon CloudWatch によるメトリクスのモニタリング
- Amazon S3 イベント通知

モニタリングツール

AWS では、Amazon S3 のモニタリングに使用できるさまざまなツールを提供しています。これらのツールの中には、自動モニタリングを設定できるものもあれば、手操作を必要とするものもあります。モニタリングタスクをできるだけ自動化することをお勧めします。

自動モニタリングツール

以下の自動化されたモニタリングツールを使用して、Amazon S3 をモニタリングし、問題が発生したときにレポートできます。

- Amazon CloudWatch のアラーム – 単一のメトリクスを指定した期間モニタリングし、特定の閾値に対する複数の期間にわたるメトリクスの値に基づいて、1 つ以上のアクションを実行します。アクションは、Amazon Simple Notification Service (Amazon SNS) のトピックまたは Amazon EC2 Auto Scaling のポリシーに送信される通知です。CloudWatch アラームは、特定の状態にあるという理由だけではアクションを呼び出しません。状態が変わって、変わった状態が指定期間にわたって維持される必要があります。詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。
- AWS CloudTrail のログのモニタリング – アカウント間でログファイルを共有し、CloudTrail のログファイルを CloudWatch Logs に送信してリアルタイムでモニタリングします。また、ログを処理するアプリケーションを Java で作成し、CloudTrail からの提供後にログファイルが変更されていないことを確認します。詳細については、「AWS CloudTrail を使用した Amazon S3 API コールのログ記録」を参照してください。

手動モニタリングツール

Amazon S3 のモニタリングでもう 1 つ重要な点は、CloudWatch のアラームのターゲット外の項目を手動でモニタリングすることです。Amazon S3、CloudWatch、Trusted Advisor、その他の AWS Management Consoleのダッシュボードには、AWS 環境の状態が一目でわかるビューが表示されます。サーバーアクセスのログ記録を有効にして、バケットへのアクセスを求めるリクエストを追跡できます。各アクセスログレコードには、1 つのアクセスリクエストに関する詳細が含まれます。内容は、リクエスタ、バケット名、リクエスト時刻、リクエストアクション、応答ステータス、およびエラーコード (存在する場合) です。詳細については、「サーバーアクセスログによるリクエストのログ記録」を参照してください。

- Amazon S3 ダッシュボードには、次の情報が表示されます。
  - 含まれるバケット、オブジェクト、およびプロパティ。

- CloudWatch のホームページでは以下について確認できます。
  - 現在のアラームとステータス
  - アラームとリソースのグラフ
  - サービスのヘルスステータス

  また、CloudWatch を使用して以下のことを行えます。
  - 重要なサービスをモニタリングするためにカスタマイズされたダッシュボードを作成する。
  - メトリクスデータをグラフ化して、問題のトラブルシューティングを行い、傾向を確認する。
  - AWS リソースのすべてのメトリクスを検索およびブラウズする。
  - 問題があることを通知するアラームを作成/編集する。
- AWS Trusted Advisor は、AWS リソースのパフォーマンス、信頼性、セキュリティ、費用効率を向上するためのモニタリングに役立ちます。すべてのユーザーは、4 つの Trusted Advisor; チェックを利用できます。ビジネスまたはエンタープライズサポートプランのユーザーは、50 以上のチェックを利用できます。詳細については、「AWS Trusted Advisor」を参照してください。

  Trusted Advisor には Amazon S3 に関連する以下のチェックがあります。
  - Amazon S3 バケットのログ記録設定をチェックします。
  - オープンなアクセス許可がある Amazon S3 バケットのセキュリティチェック。
  - バージョニングが有効になっていない、またはバージョニングが停止されている Amazon S3 バケットの耐障害性チェック。

Amazon S3 のログ記録オプション

ユーザー、ロール、または AWS のサービス よって実行されたアクションを Amazon S3 リソースに記録し、監査およびコンプライアンス目的でログレコードを管理することができます。これを行うには、サーバーアクセスのログ記録、AWS CloudTrail ログ記録、またはその両方を組み合わせて使用します。Amazon S3 リソースのバケットレベルおよびオブジェクトレベルのアクションをログ記録するには、CloudTrail を使用することをお勧めします。以下のセクションに各オプションの詳細を示します。

- サーバーアクセスログによるリクエストのログ記録
- AWS CloudTrail を使用した Amazon S3 API コールのログ記録

CloudTrail ログおよび Amazon S3 サーバーアクセスログの主なプロパティを次の表に示します。テーブルとメモを確認し、CloudTrail がセキュリティ要件を満たしていることを確認してください。

ログのプロパティ	AWS CloudTrail	Amazon S3 サーバーログ
他のシステム (Amazon CloudWatch Logs、Amazon CloudWatch Events) に転送可能	あり	No
ログを複数の宛先に配信する (例えば、同じログを 2 つの異なるバケットに送信する)	あり	No
オブジェクトのサブセット (プレフィックス) のログを有効にする	あり	No
クロスアカウントログ配信 (異なるアカウントが所有するターゲットバケットとソースバケット)	あり	No
デジタル署名またはハッシュを使用したログファイルの整合性の検証	あり	No
ログファイルの暗号化 (デフォルトまたはカスタム)	あり	No
オブジェクトオペレーション (Amazon S3 API を使用)	あり	あり
バケットオペレーション (Amazon S3 API を使用)	あり	あり
ログの検索可能な UI	あり	No
オブジェクトロックパラメータのフィールド。ログ記録用	あり	No

ログのプロパティ	AWS CloudTrail	Amazon S3 サーバーログ
の Amazon S3 Select プロパティ		
ログレコードの `Object Size`、`Total Time`、`Turn-Around Time` 、`HTTP Referer` の各フィールド	なし	あり
ライフサイクルの移行、失効、復元	なし	あり
バッチ削除オペレーションでのキーのログ記録	なし	あり
認証の失敗 1	なし	あり
ログが配信されるアカウント	バケット所有者 [2]、リクエスタ	バケット所有者名のみ
Performance and Cost	AWS CloudTrail	Amazon S3 Server Logs
価格	管理イベント (初回配信) は無料です。データイベントには料金がかかります (ログの保存は別料金)。	ログの保存にかかる料金以外の追加コストなし
ログ配信の速度	データイベント (5 分ごと)、管理イベント (15 分ごと)	数時間以内
ログ形式	JSON	スペース区切りの改行区切りレコードを含むログファイル

メモ

1. CloudTrail では、認証に失敗したリクエスト (提供された認証情報が無効なリクエスト) のログは配信されません。ただし、承認に失敗したリクエスト (`AccessDenied`) および匿名ユーザーによるリクエストのログは含まれます。

2. リクエスト内のオブジェクトへのフルアクセス権がそのアカウントに付与されていない場合、S3 バケット所有者に CloudTrail ログが送信されます。詳細については、「クロスアカウントのシナリオでの Amazon S3 オブジェクトレベルのアクション」を参照してください。
3. S3 は、VPC エンドポイントポリシーで拒否されている場合、または VPC ポリシーが評価される前にリクエストが失敗した場合、VPC エンドポイントリクエストの CloudTrail ログまたはサーバーアクセスログのリクエスタまたはバケット所有者への配信をサポートしません。

AWS CloudTrail を使用した Amazon S3 API コールのログ記録

Amazon S3 は、ユーザー、ロール、または AWS のサービス が実行したアクションの記録を提供するサービスである AWS CloudTrail と統合されています。CloudTrail は、Amazon S3 へのすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、Amazon S3 コンソールからの呼び出しと、Amazon S3 API オペレーションへのコード呼び出しが含まれます。CloudTrail で収集された情報を使用して、Amazon S3 に対するリクエスト、リクエスト元の IP アドレス、リクエストの作成日時、その他の詳細を確認できます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます:

- ルートユーザーまたはユーザー認証情報のどちらを使用してリクエストが送信されたか
- リクエストが IAM Identity Center ユーザーに代わって行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービス によって送信されたかどうか。

アカウントを作成すると、AWS アカウント で CloudTrail がアクティブになり、自動的に CloudTrail の[イベント履歴] にアクセスできるようになります。CloudTrail の [イベント履歴] では、AWS リージョン で過去 90 日間に記録された 管理イベントの表示、検索、およびダウンロードが可能で、変更不可能な記録を確認できます。詳細については、「AWS CloudTrail ユーザーガイド」の「CloudTrail イベント履歴の使用」を参照してください。[イベント履歴] の閲覧には CloudTrail の料金はかかりません。

AWS アカウント で過去 90 日間のイベントを継続的に記録するには、証跡または CloudTrail Lake イベントデータストアを作成します。

CloudTrail 証跡

証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。AWS Management Console を使用して作成した証跡はマルチリージョンです。AWS CLI を使用する際は、単一リージョンまたは複数リージョンの証跡を作成できます。アカウント内のすべて AWS リージョン でアクティビティを把握するため、マルチリージョン証跡を作成することをお勧めします。単一リージョンの証跡を作成する場合、証跡の AWS リージョン に記録されたイベントのみを表示できます。証跡の詳細については、「AWS CloudTrail ユーザーガイド」の「AWS アカウント の証跡の作成」および「組織の証跡の作成」を参照してください。

証跡を作成すると、進行中の管理イベントのコピーを 1 つ無料で CloudTrail から Amazon S3 バケットに配信できますが、Amazon S3 ストレージには料金がかかります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。Amazon S3 の料金に関する詳細については、「Amazon S3 の料金」を参照してください。

CloudTrail Lake イベントデータストア

CloudTrail Lake を使用すると、イベントに対して SQL ベースのクエリを実行できます。CloudTrail Lake は、行ベースの JSON 形式の既存のイベントを Apache ORC 形式に変換します。ORC は、データを高速に取得するために最適化された単票ストレージ形式です。イベントはイベントデータストアに集約されます。イベントデータストアは、高度なイベントセレクタを適用することによって選択する条件に基いた、イベントのイミュータブルなコレクションです。どのイベントが存続し、クエリに使用できるかは、イベントデータストアに適用するセレクタが制御します。CloudTrail Lake の詳細については、「AWS CloudTrail ユーザーガイド」の「Lake の使用AWS CloudTrail」を参照してください。

CloudTrail Lake のイベントデータストアとクエリにはコストがかかります。イベントデータストアを作成する際に、イベントデータストアに使用する料金オプションを選択します。料金オプションによって、イベントの取り込みと保存にかかる料金、および、そのイベントデータストアのデフォルトと最長の保持期間が決まります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。

必要な場合はログファイルを自身のバケットに保管できますが、ログファイルを自動的にアーカイブまたは削除するように Amazon S3 ライフサイクルルールを定義することもできます。デフォルトでは Amazon S3 のサーバー側の暗号化 (SSE) を使用して、ログファイルが暗号化されます。

CloudTrail ログを Amazon S3 サーバーアクセスログと CloudWatch Logs と併用する

AWS CloudTrail ログは、Amazon S3 のユーザー、ロール、または AWS のサービスによって実行されたアクションの記録を提供するのに対し、Amazon S3 サーバーアクセスログは、S3 バケットに対して行われたリクエストの詳細な記録を提供します。ログ別の機能とプロパティ、パフォーマンス、コストの詳細については、the section called “ログ記録オプション” を参照してください。

AWS CloudTrail ログは、Amazon S3 のサーバーアクセスログと一緒に使用できます。CloudTrail ログを使用すると、Amazon S3 バケットレベルおよびオブジェクトレベルのオペレーションの詳細な API トラッキングが提供されます。Amazon S3 のサーバーアクセスログでは、Amazon S3 内のオブジェクトレベルのデータオペレーションが可視化されます。サーバーアクセスログの詳細については、サーバーアクセスログによるリクエストのログ記録 を参照してください。

CloudTrail ログは、Amazon S3 の Amazon CloudWatch と一緒に使用することもできます。CloudTrail を CloudWatch Logs と統合すると、CloudTrail が取得した S3 バケットレベル API アクティビティが、指定した CloudWatch ロググループの CloudWatch ログストリームに送られます。特定の API アクティビティをモニタリングする CloudWatch アラームを作成し、その API アクティビティが発生した時に電子メールの通知を受け取ることができます。特定の API アクティビティをモニタリングするための CloudWatch アラームの詳細については、AWS CloudTrail ユーザーガイドを参照してください。Amazon S3 と CloudWatch の併用の詳細については、Amazon CloudWatch によるメトリクスのモニタリング を参照してください。

> **Note**
>
> S3 は、VPC エンドポイントポリシーで拒否されている場合、VPC エンドポイントリクエストのリクエスタまたはバケット所有者への CloudTrail ログの配信をサポートしません。

Amazon S3 SOAP API コールを使用した CloudTrail トラッキング

CloudTrail は、Amazon S3 SOAP API コールを追跡します。Amazon S3 SOAP のサポートは HTTP 経由で非推奨ですが、HTTPS 経由では引き続き利用可能です。Amazon S3 の SOAP に対するサポートの詳細については、付録 A: SOAP API の使用 を参照してください。

> **Important**
>
> 新しい Amazon S3 機能は、SOAP ではサポートされません。REST API か AWS SDK を使用することをお勧めします。

CloudTrail のログ記録によって追跡される Amazon S3 SOAP アクション

SOAP API 名	CloudTrail ログで使用される API イベント名
ListAllMyBuckets	`ListBuckets`
CreateBucket	`CreateBucket`
DeleteBucket	`DeleteBucket`
GetBucketAccessControlPolicy	`GetBucketAcl`
SetBucketAccessControlPolicy	`PutBucketAcl`
GetBucketLoggingStatus	`GetBucketLogging`
SetBucketLoggingStatus	`PutBucketLogging`

CloudTrail と Amazon S3 の詳細については、以下のトピックを参照してください。

トピック

- Amazon S3 CloudTrail イベント
- Simple Storage Service (Amazon S3) と S3 on Outposts の CloudTrail ログファイルエントリ
- S3 バケットとオブジェクトの CloudTrail イベントログ記録の有効化
- CloudTrail を使用した Amazon S3 リクエストの識別

Amazon S3 CloudTrail イベント

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるように

なりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

このセクションでは、S3 が CloudTrail に記録するイベントについて説明します。

CloudTrail の Amazon S3 データイベント

データイベントでは、リソース上またはリソース内で実行されるリソースオペレーション (Amazon S3 オブジェクトの読み取りまたは書き込みなど) についての情報が得られます。これらのイベントは、データプレーンオペレーションとも呼ばれます。データイベントは、多くの場合、高ボリュームのアクティビティです。デフォルトでは、CloudTrail はデータイベントをログ記録しません。CloudTrail [イベント履歴] にはデータイベントは記録されません。

追加の変更がイベントデータに適用されます。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。

CloudTrail コンソール、AWS CLI、または CloudTrail API オペレーションを使用して、Amazon S3 リソースタイプのデータイベントをログ記録できます。データイベントをログに記録する方法の詳細については、「 AWS CloudTrailユーザーガイド」の「 AWS Management Consoleを使用したデータイベントのログ記録」および「AWS Command Line Interfaceを使用したデータイベントのログ記録」を参照してください。

次の表に、データイベントをログに記録できる Amazon S3 リソースタイプを示します。データイベントタイプ (コンソール) 列には、CloudTrail コンソールの[データイベントタイプ]リストから選択する値が表示されます。resources.type 値列には、AWS CLI または CloudTrail API を使用して高度なイベントセレクタを設定するときに指定する `resources.type` 値が表示されます。CloudTrail に記録されたデータ API 列には、リソース タイプの CloudTrail にログ記録された API コールが表示されます。

データイベントタイプ (コンソール)	resources.type 値	CloudTrail にログ記録されたデータ API
S3	`AWS::S3::Object`	• AbortMultipartUpload

データイベントタイプ (コンソール)	resources.type 値	CloudTrail にログ記録されたデータ API
		• CompleteMultipartUpload • CopyObject • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • DeleteObjects • GetObject • GetObjectAcl • GetObjectAttributes • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • GetObjectTorrent • HeadObject • ListMultipartUploads • ListObjectVersions • ListObjects • ListParts • PutObject • PutObjectAcl • PutObjectLegalHold • PutObjectRetention • PutObjectTagging • RestoreObject • SelectObjectContent • UploadPart • UploadPartCopy

データイベントタイプ (コンソール)	resources.type 値	CloudTrail にログ記録されたデータ API
S3 アクセスポイント	`AWS::S3::Access Point`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject (同じリージョンへのコピーのみ) • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetBucketAcl • GetBucketCors • GetBucketLocation • GetBucketNotificationConfiguration • GetBucketPolicy • GetObject • GetObjectAcl • GetObjectAttributes • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • HeadBucket • HeadObject • ListMultipartUploads • ListObjects • ListObjectsV2 • ListObjectVersions • ListParts • Presign • PutObject

データイベントタイプ (コンソール)	resources.type 値	CloudTrail にログ記録されたデータ API
		• PutObjectLegalHold • PutObjectRetention • PutObjectAcl • PutObjectTagging • RestoreObject • UploadPart • UploadPartCopy (同じリージョンへのコピーのみ)

データイベントタイプ (コンソール)	resources.type 値	CloudTrail にログ記録されたデータ API
S3 Object Lambda	`AWS::S3ObjectLambda::AccessPoint`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject (同じリージョンへのコピーのみ) • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetObject • GetObjectAcl • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • HeadObject • ListMultipartUploads • ListObjects • ListObjectVersions • ListParts • PutObject • PutObjectLegalHold • PutObjectRetention • PutObjectAcl • PutObjectTagging • RestoreObject • UploadPart • WriteGetObjectResponse

データイベントタイプ (コンソール)	resources.type 値	CloudTrail にログ記録されたデータ API
S3 Outposts	`AWS::S3Outposts::Object`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject (同じリージョンへのコピーのみ) • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetObject • GetObjectTagging • HeadObject • ListMultipartUploads • ListObjects • ListObjectsV2 • ListParts • PutObject • PutObjectTagging • UploadPart • UploadPartCopy

`eventName`、`readOnly`、および `resources.ARN` フィールドでフィルタリングして、自分にとって重要なイベントのみをログに記録するように高度なイベントセレクタを設定できます。オブジェクトの詳細については、「AWS CloudTrail API リファレンス」の「AdvancedFieldSelector」を参照してください。

CloudTrail の Amazon S3 管理イベント

Amazon S3 は、すべてのコントロールプレーンオペレーションを管理イベントとして記録します。S3 API オペレーションの詳細については、「Amazon S3 API Reference」を参照してください。

CloudTrail が Amazon S3 に対して行われたリクエストをキャプチャする方法

デフォルトでは、CloudTrail は過去 90 日間の S3 バケットレベルの API コールをログに記録しますが、オブジェクトに対して行われたログリクエストは記録しません。バケットレベルの呼び出しには `CreateBucket`、`DeleteBucket`、`PutBucketLifecycle`、`PutBucketPolicy` などのイベントが含まれます。バケットレベルのイベントは CloudTrail コンソールで確認できますが、そこでデータイベント (Amazon S3 オブジェクトレベルのコール) を確認することはできません。それらについて CloudTrail ログを解析またはクエリする必要があります。

CloudTrail ロギングによって追跡される Amazon S3 アカウントレベルのアクション

CloudTrail はアカウントレベルのアクションを記録します。Amazon S3 レコードは、他の AWS のサービス のレコードと一緒にログファイルに記録されます。CloudTrail は、期間とファイルサイズに基づいて、新しいファイルをいつ作成して書き込むかを決定します。

このセクションでは、CloudTrail によるログ記録でサポートされている Amazon S3 アカウントレベルのアクションの一覧を示します。

CloudTrail のログ記録によって追跡される Amazon S3 アカウントレベルの API アクションは、以下のイベント名で表示されます。CloudTrail イベント名は API アクション名とは異なります。例えば、DeletePublicAccessBlock は DeleteAccountPublicAccessBlock です。

- DeleteAccountPublicAccessBlock
- GetAccountPublicAccessBlock
- PutAccountPublicAccessBlock

CloudTrail ログ記録によって追跡される Amazon S3 バケットレベルのアクション

デフォルトでは、CloudTrail はバケットレベルのアクションをログに記録します。Amazon S3 レコードは、他の AWS のサービスレコードと一緒にログファイルに記録されます。CloudTrail は、期間とファイルサイズに基づいて、新しいファイルをいつ作成して書き込むかを決定します。

このセクションには、CloudTrail によるログ記録でサポートされている Amazon S3 バケットレベルのアクションの一覧が記載されています。

CloudTrail のログ記録によって追跡される Amazon S3 バケットレベルの API アクションは、以下のイベント名で表示されます。CloudTrail イベント名が API アクション名と異なる場合があります。たとえば、`PutBucketLifecycleConfiguration` は `PutBucketLifecycle` です。

- CreateBucket
- DeleteBucket
- DeleteBucketAnalyticsConfiguration
- DeleteBucketCors
- DeleteBucketEncryption
- DeleteBucketIntelligentTieringConfiguration
- DeleteBucketInventoryConfiguration
- DeleteBucketLifecycle
- DeleteBucketMetricsConfiguration
- DeleteBucketOwnershipControls
- DeleteBucketPolicy
- DeleteBucketPublicAccessBlock
- DeleteBucketReplication
- DeleteBucketTagging
- GetAccelerateConfiguration
- GetBucketAcl
- GetBucketAnalyticsConfiguration
- GetBucketCors
- GetBucketEncryption
- GetBucketIntelligentTieringConfiguration
- GetBucketInventoryConfiguration
- GetBucketLifecycle
- GetBucketLocation
- GetBucketLogging
- GetBucketMetricsConfiguration
- GetBucketNotification
- GetBucketObjectLockConfiguration
- GetBucketOwnershipControls

- GetBucketPolicy
- GetBucketPolicyStatus
- GetBucketPublicAccessBlock
- GetBucketReplication
- GetBucketRequestPayment
- GetBucketTagging
- GetBucketVersioning
- GetBucketWebsite
- HeadBucket
- ListBuckets
- PutAccelerateConfiguration
- PutBucketAcl
- PutBucketAnalyticsConfiguration
- PutBucketCors
- PutBucketEncryption
- PutBucketIntelligentTieringConfiguration
- PutBucketInventoryConfiguration
- PutBucketLifecycle
- PutBucketLogging
- PutBucketMetricsConfiguration
- PutBucketNotification
- PutBucketObjectLockConfiguration
- PutBucketOwnershipControls
- PutBucketPolicy
- PutBucketPublicAccessBlock
- PutBucketReplication
- PutBucketRequestPayment
- PutBucketTagging
- PutBucketVersioning

- PutBucketWebsite

これらの API オペレーションに加えて、OPTIONS オブジェクトのオブジェクトレベルのアクションを使用することもできます。このアクションは、バケットの CORS 設定を確認するため、CloudTrail ログ記録のバケットレベルアクションと同様に扱われます。

CloudTrail ログ記録が追跡する S3 Express One Zone バケットレベル (リージョン API エンドポイント) アクション

デフォルトでは、CloudTrail はディレクトリバケットのバケットレベルのアクションを管理イベントとして記録します。S3 Express One Zone の CloudTrail 管理イベントの `eventsource` は、`s3express.amazonaws.com` です。

> **Note**
>
> S3 Express One Zone の場合、ゾーンエンドポイント (オブジェクトレベルまたはデータプレーン) API オペレーション (`PutObject` または `GetObject` など) の CloudTrail ログ記録はサポートされていません。

次のリージョンエンドポイント API オペレーションが CloudTrail に記録されます。

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- PutBucketPolicy
- ListDirectoryBuckets

詳細については、「S3 Express One Zone のセキュリティのベスト プラクティス」を参照してください。

クロスアカウントのシナリオでの Amazon S3 オブジェクトレベルのアクション

クロスアカウントのシナリオでオブジェクトレベルの API コールに関連する特殊なユースケースと CloudTrail ログが報告される方法を次に示します。CloudTrail は、ログエントリが編集または省略さ

れる一部のアクセス拒否ケースを除き、リクエスタ (API コールを行ったアカウント) にログを配信します。クロスアカウントアクセスを設定する際には、このセクションの例を検討します。

> **Note**
>
> 例では、CloudTrail ログが適切に設定されていることを前提とします。

例 1: CloudTrail がバケット所有者にログを配信する

CloudTrail は、バケット所有者が同じオブジェクト API オペレーションに対するアクセス許可を持たない場合でも、バケット所有者にログを配信します。次のクロスアカウントのシナリオを検討してください。

- アカウント A がバケットを所有しています。
- アカウント B (リクエスタ) が、そのバケット内のオブジェクトへのアクセスを試みます。
- アカウント C がオブジェクトを所有しています。アカウント C はアカウント A と同じアカウントである場合とそうでない場合があります。

> **Note**
>
> CloudTrail は、常にオブジェクトレベルの API ログをリクエスタ (アカウント B) に配信します。さらに、CloudTrail は、バケット所有者 (アカウント A) がそのオブジェクト (アカウント C) を所有していないか、そのオブジェクトに対する同じ API オペレーションへのアクセス許可がない場合でも、バケット所有者に同じログを配信します。

例 2: CloudTrail は設定オブジェクト ACL で使用されている E メールアドレスを拡散しない

次のクロスアカウントのシナリオを検討してください。

- アカウント A がバケットを所有しています。
- アカウント B (リクエスタ) は、E メールアドレスを使用してオブジェクト ACL の付与を設定するためのリクエストを送信します。ACL の詳細については、アクセスコントロールリスト (ACL) の概要 を参照してください。

リクエスタは E メール情報と共にログを取得します。ただし、例 1 のようにバケット所有者がログの受信資格を持つ場合、バケット所有者はイベントを報告する CloudTrail ログを取得します。一

方、バケット所有者は ACL 設定情報 (具体的には被付与者の E メールと権限) を取得しません。ログがバケット所有者に通知する情報は、アカウント B によって ACL API コールが行われたということだけです。

Simple Storage Service (Amazon S3) と S3 on Outposts の CloudTrail ログファイルエントリ

> **Important**
>
> Amazon S3 では、Amazon S3 内のすべてのバケットの基本レベルの暗号化として、Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルト暗号化に関するよくある質問」を参照してください。

各イベントは任意の送信元からの単一のリクエストを表し、リクエストされた API オペレーション、オペレーションの日時、リクエストパラメータなどに関する情報を含みます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、イベントは特定の順序で表示されません。

詳細については、以下の例を参照してください。

トピック

- 例: Amazon S3 の CloudTrail ログファイルのエントリ
- 例: Amazon S3 on Outposts のログファイルのエントリ

例: Amazon S3 の CloudTrail ログファイルのエントリ

次は、GET サービス、PutBucketAcl および GetBucketVersioning のアクションを示す CloudTrail ログエントリの例です。

```
{
    "Records": [
```

```
    {
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
        "eventTime": "2019-02-01T03:18:19Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "ListBuckets",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "[]",
        "requestParameters": {
            "host": [
                "s3.us-west-2.amazonaws.com"
            ]
        },
        "responseElements": null,
        "additionalEventData": {
            "SignatureVersion": "SigV2",
            "AuthenticationMethod": "QueryString",
            "aclRequired": "Yes"
    },
        "requestID": "47B8E8D397DCE7A6",
        "eventID": "cdc4b7ed-e171-4cef-975a-ad829d4123e8",
        "eventType": "AwsApiCall",
        "recipientAccountId": "444455556666",
        "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    },
    {
       "eventVersion": "1.03",
       "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
```

```
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
      "eventTime": "2019-02-01T03:22:33Z",
      "eventSource": "s3.amazonaws.com",
      "eventName": "PutBucketAcl",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "",
      "userAgent": "[]",
      "requestParameters": {
          "bucketName": "",
          "AccessControlPolicy": {
              "AccessControlList": {
                  "Grant": {
                      "Grantee": {
                          "xsi:type": "CanonicalUser",
                          "xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
                          "ID":
"d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
                       },
                      "Permission": "FULL_CONTROL"
                   }
              },
              "xmlns": "http://s3.amazonaws.com/doc/2006-03-01/",
              "Owner": {
                  "ID":
"d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
              }
          },
          "host": [
              "s3.us-west-2.amazonaws.com"
          ],
          "acl": [
              ""
          ]
      },
      "responseElements": null,
      "additionalEventData": {
          "SignatureVersion": "SigV4",
          "CipherSuite": "ECDHE-RSA-AES128-SHA",
          "AuthenticationMethod": "AuthHeader"
      },
      "requestID": "BD8798EACDD16751",
      "eventID": "607b9532-1423-41c7-b048-ec2641693c47",
```

```
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333",
      "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    },
    {
      "eventVersion": "1.03",
      "userIdentity": {
          "type": "IAMUser",
          "principalId": "111122223333",
          "arn": "arn:aws:iam::111122223333:user/myUserName",
          "accountId": "111122223333",
          "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
          "userName": "myUserName"
        },
      "eventTime": "2019-02-01T03:26:37Z",
      "eventSource": "s3.amazonaws.com",
      "eventName": "GetBucketVersioning",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "",
      "userAgent": "[]",
      "requestParameters": {
          "host": [
              "s3.us-west-2.amazonaws.com"
          ],
          "bucketName": "example-s3-bucket1",
          "versioning": [
              ""
          ]
      },
      "responseElements": null,
      "additionalEventData": {
          "SignatureVersion": "SigV4",
          "CipherSuite": "ECDHE-RSA-AES128-SHA",
          "AuthenticationMethod": "AuthHeader"
    },
      "requestID": "07D681279BD94AED",
      "eventID": "f2b287f3-0df1-4961-a2f4-c4bdfed47657",
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333",
      "tlsDetails": {
```

```
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    }
  ]
}
```

例: Amazon S3 on Outposts のログファイルのエントリ

Amazon S3 on Outposts 管理イベントは、AWS CloudTrail を通じて利用できます。詳細については、「AWS CloudTrail を使用した Amazon S3 API コールのログ記録」を参照してください。さらに、必要に応じてAWS CloudTrail のデータイベントのログ記録を有効にすることもできます。

証跡は、リージョン内の指定した S3 バケットにイベントをログファイルとして配信するように設定できます。Outposts バケットの CloudTrail ログには新しいフィールドの `edgeDeviceDetails` が含まれており、指定されたバケットが配置されている Outposts を識別できます。

追加のログフィールドには、要求されたアクション、アクションの日時、リクエストパラメータが含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例では、`s3-outposts` での PutObject アクションを示す CloudTrail ログエントリについて説明します。

```
{
      "eventVersion": "1.08",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "111122223333",
        "arn": "arn:aws:iam::111122223333:user/yourUserName",
        "accountId": "222222222222",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "yourUserName"
      },
      "eventTime": "2020-11-30T15:44:33Z",
      "eventSource": "s3-outposts.amazonaws.com",
      "eventName": "PutObject",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "26.29.66.20",
      "userAgent": "aws-cli/1.18.39 Python/3.4.10 Darwin/18.7.0 botocore/1.15.39",
      "requestParameters": {
```

```
        "expires": "Wed, 21 Oct 2020 07:28:00 GMT",
        "Content-Language": "english",
        "x-amz-server-side-encryption-customer-key-MD5": "wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
        "ObjectCannedACL": "BucketOwnerFullControl",
        "x-amz-server-side-encryption": "Aes256",
        "Content-Encoding": "gzip",
        "Content-Length": "10",
        "Cache-Control": "no-cache",
        "Content-Type": "text/html; charset=UTF-8",
        "Content-Disposition": "attachment",
        "Content-MD5": "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY",
        "x-amz-storage-class": "Outposts",
        "x-amz-server-side-encryption-customer-algorithm": "Aes256",
        "bucketName": "example-s3-bucket1",
        "Key": "path/upload.sh"
      },
      "responseElements": {
        "x-amz-server-side-encryption-customer-key-MD5": "wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
        "x-amz-server-side-encryption": "Aes256",
        "x-amz-version-id": "001",
        "x-amz-server-side-encryption-customer-algorithm": "Aes256",
        "ETag": "d41d8cd98f00b204e9800998ecf8427f"
      },
      "additionalEventData": {
        "CipherSuite": "ECDHE-RSA-AES128-SHA",
        "bytesTransferredIn": 10,
        "x-amz-id-2": "29xXQBV20
+xOHKItvzY1suLv1i6A52E0zOX159fpfsItYd58JhXwKxXAXI4IQkp6",
        "SignatureVersion": "SigV4",
        "bytesTransferredOut": 20,
        "AuthenticationMethod": "AuthHeader"
      },
      "requestID": "8E96D972160306FA",
      "eventID": "ee3b4e0c-ab12-459b-9998-0a5a6f2e4015",
      "readOnly": false,
      "resources": [
        {
          "accountId": "222222222222",
          "type": "AWS::S3Outposts::Object",
          "ARN": "arn:aws:s3-outposts:us-east-1:YYY:outpost/op-01ac5d28a6a232904/
bucket/path/upload.sh"
        },
```

```
        {
          "accountId": "222222222222",
          "type": "AWS::S3Outposts::Bucket",
          "ARN": "arn:aws:s3-outposts:us-east-1:YYY:outpost/op-01ac5d28a6a232904/
bucket/"
        }
      ],
      "eventType": "AwsApiCall",
      "managementEvent": false,
      "recipientAccountId": "444455556666",
      "sharedEventID": "02759a4c-c040-4758-b84b-7cbaaf17747a",
      "edgeDeviceDetails": {
        "type": "outposts",
        "deviceId": "op-01ac5d28a6a232904"
      },
      "eventCategory": "Data"
    }
```

S3 バケットとオブジェクトの CloudTrail イベントログ記録の有効化

CloudTrail データイベントを使用して、Amazon S3 のバケットおよびオブジェクトレベルのリクエストに関する情報を取得できます。すべてのバケットまたは特定のバケットのリストの CloudTrail データイベントを有効にするには、CloudTrail で手動で証跡を作成する必要があります。

> **Note**
>
> - CloudTrail のデフォルト設定では、管理イベントのみが検出されます。アカウントに対してデータイベントを有効にしていることをチェックして確認します。
> - 高いワークロードを生成している S3 バケットがある場合、短期間で数千のログを生成する可能性があります。ビジーなバケットでは、CloudTrail データイベントを有効にする期間にご注意ください。

CloudTrail は、選択した S3 バケットに Amazon S3 データイベントを保存します。クエリと分析を簡素化するには、所有する複数のバケットのイベントを適切にまとめられるように別の AWS アカウント のバケットの使用を検討することをお勧めします。AWS Organizations では、モニタリングしているバケットを所有しているアカウントにリンクされた AWS アカウント を簡単に作成することができます。詳細については、『AWS Organizations ユーザーガイド』の「What is AWS Organizations? (とは ?)」 を参照してください。

CloudTrail で証跡のデータイベントをログに記録する場合、アドバンストイベントセレクタまたはベーシックイベントセレクタのどちらを使用するかを選択できます。アドバンストイベントセレクタを使用して CloudTrail コンソールで証跡を作成する場合、データイベントセクションで、[ログセレクタテンプレート] の [すべてのイベントをログに記録する] を選択して、すべてのオブジェクトレベルのイベントをログ記録できます。ベーシックイベントセレクタを使用して CloudTrail コンソールで証跡を作成する場合、データイベントセクションで、[アカウントのすべての S3 バケットを選択する] チェックボックスを選択して、すべてのオブジェクトレベルイベントをログ記録できます。

> **Note**
>
> - ベストプラクティスとして、AWS CloudTrail データイベントバケットに対してライフサイクル設定を作成することをお勧めします。ログファイルを監査する必要があると考える期間が経過したらログファイルを定期的に削除するように、ライフサイクル設定を設定します。これにより、各クエリで Athena が分析するデータの量が減ります。詳細については、「バケットにライフサイクル設定を設定する」を参照してください。
> - ロギング形式の詳細については、AWS CloudTrail を使用した Amazon S3 API コールのログ記録 を参照してください。
> - CloudTrail ログをクエリする方法の例については、AWS ビッグデータブログの記事 Analyze Security, Compliance, and Operational Activity Using AWS CloudTrail and Amazon Athena を参照してください。

コンソールを使用してバケット内のオブジェクトのログ記録を有効にする

Amazon S3 コンソールを使用し、AWS CloudTrail 証跡を設定して、S3 バケット内のオブジェクトのデータイベントをログに記録できます。CloudTrail では、`GetObject`、`DeleteObject`、`PutObject` など、Amazon S3 オブジェクトレベルの API オペレーションのログ記録がサポートされます。これらのイベントは、データイベントと呼ばれます。

デフォルトでは CloudTrail 証跡はデータイベントを記録しませんが、証跡を設定して、指定した S3 バケットのデータイベントを記録するか、AWS アカウントですべての Amazon S3 バケットのデータイベントを記録するようにできます。詳細については、「AWS CloudTrail を使用した Amazon S3 API コールのログ記録」を参照してください。

CloudTrail では、CloudTrail イベント履歴にデータイベントが設定されません。さらに、すべてのバケットレベルのアクションが CloudTrail イベント履歴に入力されるわけではありません。CloudTrail のログ記録によって追跡される Amazon S3 バケットレベルの API アクションの詳細については、

「CloudTrail ログ記録によって追跡される Amazon S3 バケットレベルのアクション」を参照してください。CloudTrail ログのクエリ方法の詳細については、Amazon CloudWatch Logs のフィルタパターンと Amazon Athena を使用した CloudTrail ログのクエリに関する AWS ナレッジセンターの記事を参照してください。

S3 バケットのデータイベントをログに記録するように証跡を設定する場合、AWS CloudTrail コンソールまたは Amazon S3 コンソールのいずれかを使用できます。AWS アカウント内のすべての Amazon S3 バケットのデータイベントを記録するように証跡を設定する場合は、CloudTrail コンソールを使用する方が簡単です。CloudTrail コンソールを使用して S3 データイベントを記録するように証跡を設定する方法については、AWS CloudTrail ユーザーガイドのデータイベントを参照してください。

> **Important**
>
> 追加の変更がイベントデータに適用されます。詳細については、AWS CloudTrail 料金表を参照してください。

以下の手順では、Amazon S3 コンソールを使用して、CloudTrail の証跡で S3 バケットのデータイベントの記録を設定する方法を示します。

S3 バケットでオブジェクトの CloudTrail データイベントの記録を有効にするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット名] リストで、バケットの名前を選択します。
3. [プロパティ] を選択します。
4. AWS CloudTrailデータイベントで、CloudTrail の設定を選択します。

   新しい CloudTrail の証跡を作成するか、既存の証跡を再利用して、証跡に記録するように Amazon S3 データイベントを設定できます。CloudTrail コンソールで証跡を作成する方法については、AWS CloudTrail ユーザーガイドのコンソールで証跡を作成および更新するを参照してください。CloudTrail コンソールで Amazon S3 データイベントロギングを設定する方法については、AWS CloudTrail ユーザーガイドの Amazon S3 オブジェクトのデータイベントをログに記録するを参照してください。

> **Note**
>
> S3 バケットのデータイベントをログに記録するための証跡の設定に CloudTrail コンソールまたは Amazon S3 コンソールを使用する場合、そのバケットに対してオブジェクトレベルのロギングが有効化されていることが Amazon S3 コンソールに表示されます。

S3 バケットでオブジェクトの CloudTrail データイベントの記録を無効にするには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudtrail/で CloudTrail コンソールを開きます。
2. 左のナビゲーションペインで、[証跡] を選択します。
3. バケットのイベントを記録するために作成した証跡の名前を選択します。
4. 証跡の詳細ページで、右上隅にある [ロギングを停止] を選択します。
5. 表示されたダイアログボックスで [ロギングを停止] を選択します。

S3 バケットを作成するときにオブジェクトレベルのログ記録を有効にする方法の詳細については、バケットの作成 を参照してください。

S3 バケットを使用した CloudTrail ログ記録の詳細については、以下のトピックを参照してください。

- S3 バケットのプロパティを表示するには
- AWS CloudTrail を使用した Amazon S3 API コールのログ記録
- AWS CloudTrail ユーザーガイドの CloudTrail ログファイルの操作

CloudTrail を使用した Amazon S3 リクエストの識別

Amazon S3 では、AWS CloudTrail イベントログを使用してリクエストを識別することができます。AWS CloudTrail は、Amazon S3 リクエストを識別する上で推奨される方法ですが、Amazon S3 サーバーアクセスログを使用している場合は、the section called “S3 リクエストの識別” を参照してください。

トピック

- CloudTrail ログで Amazon S3 に対して行われたリクエストを特定する
- CloudTrail を使用した Amazon S3 Signature Version 2 リクエストの識別
- CloudTrail を使用した S3 オブジェクトへのアクセスの識別

CloudTrail ログで Amazon S3 に対して行われたリクエストを特定する

バケットにイベントを送信するために CloudTrail をセットアップした後、Amazon S3 コンソールで送信先のバケットにオブジェクトが送信されるのを確認できるようになります。これらは次の形式になっています。

```
s3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/Region/yyyy/mm/dd
```

CloudTrail により ログ記録されたイベントは、S3 バケットにある圧縮された gzipped JSON オブジェクトに保存されます。リクエストを効率的に見つけるには、Amazon Athena などのサービスを利用して CloudTrail ログにインデックスを作成し、クエリします。

CloudTrail と Athena の詳細については、「Amazon Athena ユーザーガイド」の「パーティション射影を使用した Athena での AWS CloudTrail ログ用のテーブルの作成」を参照してください。

CloudTrail を使用した Amazon S3 Signature Version 2 リクエストの識別

CloudTrail イベントログを使用して、Amazon S3 でリクエストに署名するために使用された API 署名バージョンを特定できます。署名バージョン 2 のサポートがオフになる (廃止される) ため、この機能は重要です。その後、Amazon S3 は署名バージョン 2 を使用するリクエストを受け入れず、すべてのリクエストに Signature Version 4 署名を使用する必要があります。

CloudTrail を使用して、署名バージョン 2 の署名を使用しているワークフローがあるかどうかを確認することを強くお勧めします。業務に影響が出ないように、Signature Version 4 を使用してライブラリとコードをアップグレードし、修正します。

詳細については、AWS re:Post の「お知らせ: Amazon S3 用の AWS CloudTrail は、セキュリティ監査を強化するための新しいフィールドを追加します」を参照してください。

> **Note**
>
> Amazon S3 の CloudTrail イベントには、キー名である「`additionalEventData`」のリクエスト詳細に署名バージョンが含まれます。Amazon S3 にあるオブジェクトに対する `GET`、`PUT`、`DELETE` などのリクエストで Signature Version を見つけるには、CloudTrail

データイベントを有効にする必要があります。(デフォルトでは、この機能はオフになっています。)

Signature Version 2 リクエストを識別する方法として、AWS CloudTrail が推奨されます。Amazon S3 のサーバーアクセスログを使用している場合は、「Amazon S3 アクセスログを使用した Signature Version 2 リクエストの識別」を参照してください。

トピック

- Amazon S3 署名バージョン 2 リクエストを識別する Athena クエリの例
- 署名バージョン 2 データのパーティション化

Amazon S3 署名バージョン 2 リクエストを識別する Athena クエリの例

Example — Signature Version 2 のイベントをすべて選択し、**EventTime**、**S3_Action**、**Request_Parameters**、**Region**、**SourceIP**、**UserAgent** のみを印刷する

以下の Athena クエリで、*s3_cloudtrail_events_db.cloudtrail_table* を Athena の詳細と置き換え、必要に応じて上限を引き上げるか削除します。

```
SELECT EventTime, EventName as S3_Action, requestParameters as Request_Parameters,
 awsregion as AWS_Region, sourceipaddress as Source_IP, useragent as User_Agent
FROM s3_cloudtrail_events_db.cloudtrail_table
WHERE eventsource='s3.amazonaws.com'
AND json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
LIMIT 10;
```

Example – 署名バージョン 2 トラフィックを送信しているすべてのリクエスタを選択します。

```
SELECT useridentity.arn, Count(requestid) as RequestCount
FROM s3_cloudtrail_events_db.cloudtrail_table
WHERE eventsource='s3.amazonaws.com'
    and json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
Group by useridentity.arn
```

署名バージョン 2 データのパーティション化

クエリするデータが大量にある場合、パーティション化されたテーブルを作成することにより Athena のコストを削減しランタイムを短縮できます。

それには、次のようなパーティションがある新しいテーブルを作成します。

```
CREATE EXTERNAL TABLE s3_cloudtrail_events_db.cloudtrail_table_partitioned(
     eventversion STRING,
     userIdentity STRUCT<
         type:STRING,
         principalid:STRING,
         arn:STRING,
         accountid:STRING,
         invokedby:STRING,
         accesskeyid:STRING,
         userName:STRING,
      sessioncontext:STRUCT<
                 attributes:STRUCT<
                 mfaauthenticated:STRING,
                 creationdate:STRING>,
                 sessionIssuer:STRUCT<
                 type:STRING,
                 principalId:STRING,
                 arn:STRING,
                 accountId:STRING,
                 userName:STRING>
             >
          >,
     eventTime STRING,
     eventSource STRING,
     eventName STRING,
     awsRegion STRING,
     sourceIpAddress STRING,
     userAgent STRING,
     errorCode STRING,
     errorMessage STRING,
     requestParameters STRING,
     responseElements STRING,
     additionalEventData STRING,
     requestId STRING,
     eventId STRING,
     resources ARRAY<STRUCT<ARN:STRING,accountId: STRING,type:STRING>>,
```

```
        eventType STRING,
        apiVersion STRING,
        readOnly STRING,
        recipientAccountId STRING,
        serviceEventDetails STRING,
        sharedEventID STRING,
        vpcEndpointId STRING
    )
    PARTITIONED BY (region string, year string, month string, day string)
    ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
    STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
    LOCATION 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/';
```

その後で個々にパーティションを作成します。作成していない日付から結果を取得することはできません。

```
ALTER TABLE s3_cloudtrail_events_db.cloudtrail_table_partitioned ADD
    PARTITION (region= 'us-east-1', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/us-east-1/2019/02/19/'
    PARTITION (region= 'us-west-1', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/us-west-1/2019/02/19/'
    PARTITION (region= 'us-west-2', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/us-west-2/2019/02/19/'
    PARTITION (region= 'ap-southeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/ap-southeast-1/2019/02/19/'
    PARTITION (region= 'ap-southeast-2', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/ap-southeast-2/2019/02/19/'
    PARTITION (region= 'ap-northeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/ap-northeast-1/2019/02/19/'
    PARTITION (region= 'eu-west-1', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/eu-west-1/2019/02/19/'
    PARTITION (region= 'sa-east-1', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/sa-east-1/2019/02/19/';
```

これらのパーティションに基づいてリクエストを実行でき、バケット全体を読み込む必要はありません。

```
SELECT useridentity.arn,
```

```
Count(requestid) AS RequestCount
FROM s3_cloudtrail_events_db.cloudtrail_table_partitioned
WHERE eventsource='s3.amazonaws.com'
AND json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
AND region='us-east-1'
AND year='2019'
AND month='02'
AND day='19'
Group by useridentity.arn
```

CloudTrail を使用した S3 オブジェクトへのアクセスの識別

AWS CloudTrail イベントログを使用して、GetObject、DeleteObject、PutObject などのデータイベントに対する Amazon S3 オブジェクトアクセスリクエストを識別し、それらのリクエストに関する追加情報を確認できます。

以下の例は、AWS CloudTrail イベントログから Amazon S3 のすべての PUT オブジェクトリクエストを取得する方法を示しています。

トピック

- Amazon S3 オブジェクトアクセスリクエストを識別する Athena クエリの例

Amazon S3 オブジェクトアクセスリクエストを識別する Athena クエリの例

以下の Athena クエリの例で、*s3_cloudtrail_events_db.cloudtrail_table* を Athena の詳細と置き換え、必要に応じてデータ範囲を変更します。

Example — **PUT** オブジェクトアクセス要求のあるイベントをすべて選択し、**EventTime**、**EventSource**、**SourceIP**、**UserAgent**、**BucketName**、**object**、**UserARN** のみ印刷する

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  userIdentity.arn as userArn
```

```
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  eventName = 'PutObject'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example — **GET** オブジェクトアクセス要求のあるイベントをすべて選択し、**EventTime**、**EventSource**、**SourceIP**、**UserAgent**、**BucketName**、**object**、**UserARN** のみ印刷する

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  userIdentity.arn as userArn
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  eventName = 'GetObject'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example — 特定の期間にバケットに対して行われた匿名のリクエストイベントをすべて選択し、**EventTime**、**EventName**、**EventSource**、**SourceIP**、**UserAgent**、**BucketName**、**UserARN**、**Ac** のみ印刷する

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  userIdentity.arn as userArn,
  userIdentity.accountId
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
```

```
  userIdentity.accountId = 'anonymous'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example — 承認に ACL が必要なリクエストをすべて特定する

次の Amazon Athena のクエリの例は、承認のためにアクセスコントロールリスト (ACL) が必要な S3 バケットへのすべてのリクエストを特定する方法を示しています。リクエストに承認用の ACL が必要な場合、additionalEventData の aclRequired の値は Yes です。ACL が不要な場合は、aclRequired は存在しません。この情報を使用して、これらの ACL 権限を適切なバケットポリシーに移行できます。これらのバケットポリシーを作成したら、これらのバケットの ACL を無効にできます。ACL の無効化の詳細については、「ACL を無効にする前提条件」を参照してください。

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  userIdentity.arn as userArn,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  json_extract_scalar(additionalEventData, '$.aclRequired') as aclRequired
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  json_extract_scalar(additionalEventData, '$.aclRequired') = 'Yes'
  AND eventTime BETWEEN '2022-05-10T00:00:00Z' and '2022-08-10T00:00:00Z'
```

> **Note**
>
> - このクエリの例は、セキュリティのモニタリングにも役立つ場合があります。予期しないまたは不正な IP アドレスやリクエスタからの PutObject または GetObject コールの結果を確認し、バケットへの匿名リクエストを特定できます。
> - このクエリでは、ログ記録が有効になった時間以降の情報のみ取得されます。

Amazon S3 のサーバーアクセスログを使用している場合は、Amazon S3 アクセスログを使用したオブジェクトアクセスリクエストの識別 を参照してください。

サーバーアクセスログによるリクエストのログ記録

サーバーアクセスのログには、バケットに対するリクエストの詳細が記録されます。サーバーアクセスのログは、多くのアプリケーションに役立ちます。例えば、アクセスのログ情報は、セキュリティやアクセスの監査に役立ちます。この情報は、顧客基盤の把握や Amazon S3 の請求の理解にもつながります。

> **Note**
>
> サーバーアクセスログには、2019 年 3 月 20 日以降に開設されたリージョンで発生するリージョン違いによるリダイレクトエラーに関する情報は記録されません。リージョン違いによるリダイレクトエラーは、オブジェクトまたはバケットに対するリクエストがそのバケットがあるリージョン以外で行われた場合に発生します。

ログ配信を有効にするにはどうすればよいですか?

ログ配信を有効にするには、次の基本的な手順を実行します。詳細については、Amazon S3 サーバーアクセスログを有効にします。 を参照してください。

1. 送信先バケット (別名 ターゲットバケット) の名前を指定します。このバケットは、Amazon S3 がアクセスログをオブジェクトとして保存する場所です。ソースと送信先バケットの両方が同じ AWS リージョン にあり、同じアカウントが所有している必要があります。送信先バケットには S3 オブジェクトロックのデフォルト保持期間設定を指定できません。また、送信先バケットでは [リクエスタ支払い] を有効にできません。

   ログの保存先のバケットとして、ソースバケットと同じリージョンにあるユーザー所有のバケットを指定できます。これにはソースバケット自体も含まれます。ただし、ログを管理しやすくするため、アクセスログは別のバケットに保存することをお勧めします。

   ソースバケットと送信先バケットが同じである場合、バケットに書き込まれるログに関する追加のログが作成されます。ストレージの請求額がいくらか増える可能性があるため、この方法はお勧めしていません。また、ログに関する追加のログのために、必要なログを見つけにくくなります。

   アクセス ログをソースバケットに保存する場合は、すべてのログオブジェクトキーに対して送信先プレフィックス (別名ターゲットプレフィックス) を指定することをお勧めします。プレフィックスを指定すると、すべてのログオブジェクト名が共通の文字列で始まるため、ログオブジェクトを識別しやすくなります。

2. (オプション) Amazon S3 のすべてのログオブジェクトのキーにプレフィックスを割り当てます。送信先プレフィクス (別名ターゲットプレフィックス) を使用すると、ログオブジェクトを識別しやすくなります。例えば、プレフィックスの値として `logs/` を指定すると、Amazon S3 で作成する各ログオブジェクトのキーの先頭に `logs/` というプレフィックスが付けられます。

```
logs/2013-11-01-21-32-16-E568B2907131C0C0
```

プレフィックス値 `logs` を指定すると、ログオブジェクトは次のように表示されます。

```
logs2013-11-01-21-32-16-E568B2907131C0C0
```

プレフィックス は、複数のバケットログが同じ送信先バケットにログを記録する場合にもソースバケットの識別に役立ちます。

キープレフィックスは、ログを削除する際にも役に立ちます。例えば、特定のキープレフィックスを使用して、Amazon S3 のライフサイクル設定ルールを指定することができます。詳細については、「Amazon S3 ログファイルの削除」を参照してください。

3. (オプション) 生成されたログをその他のユーザーが利用するために、アクセス許可を設定します。デフォルトでは、バケット所有者のみにログオブジェクトへのフルアクセスが許可されます。(サーバーアクセスログが保存されている) 送信先バケットが S3 オブジェクト所有権のバケット所有者強制設定を使用してアクセスコントロールリスト (ACL) を無効化する場合、ACL を使用する送信先権限のアクセス許可を付与することはできません。ただし、送信先バケットのバケットポリシーを更新して、その他のユーザーにアクセスを付与することはできます。詳細については、Amazon S3 用 Identity and Access Managementおよびログ配信許可を参照してください。
4. (オプション) ログファイルのログオブジェクトキー形式を設定します。ログオブジェクトキーの形式 (別名ターゲットオブジェクトキー形式) には 2 つのオプションがあります。
   - 日付ベース以外のパーティション分割 – これはオリジナルのログオブジェクトキーの形式です。この形式を選択すると、ログファイルのキー形式は次のとおりになります。

     ```
     [DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
     ```

     例えば、`logs/` をプレフィックスとして指定すると、ログオブジェクトには次のような名前が付けられます。

     ```
     logs/2013-11-01-21-32-16-E568B2907131C0C0
     ```

- 日付ベースのパーティション分割 – 日付ベースのパーティショニングを選択した場合、ログ形式で使用される日付ソースとして、ログファイルのイベント時間または配信時間を選択できます。この形式を使用すると、ログのクエリが簡単になります。

  日付ベースのパーティション分割を選択すると、ログファイルのキー形式は次のとおりになります。

  ```
  [DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/
  [DD]/[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
  ```

  例えば、logs/ をターゲットプレフィックスとして指定すると、ログオブジェクトには次のような名前が付けられます。

  ```
  logs/123456789012/us-west-2/DOC-EXAMPLE-SOURCE-
  BUCKET/2023/03/01/2023-03-01-21-32-16-E568B2907131C0C0
  ```

  配信時間配信の場合、ログファイル名の時刻はログファイルの配信時間に対応します。

  イベント時間配信の場合、年、月、日はイベントが発生した日に対応し、時、分、秒はキーで 00 に設定されます。このようなログファイルに配信されるログは、特定の日のみを対象とします。

AWS Command Line Interface (AWS CLI)、AWS SDK、または Amazon S3 REST API を使用してログを設定する場合は、TargetObjectKeyFormat を使用してログオブジェクトのキー形式を指定します。日付ベース以外のパーティション分割を指定するには、SimplePrefix を使用します。日付ベースのパーティション分割を指定するには、PartitionedPrefix を使用します。PartitionedPrefix を使用する場合は、PartitionDateSource で EventTime または DeliveryTime を指定します。

SimplePrefix の場合、ログファイルキー形式は、次のとおりです。

```
[TargetPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
```

イベント時刻または配信時刻の PartitionedPrefix の場合、ログファイルのキー形式は次のとおりになります。

```
[TargetPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/
[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
```

ログオブジェクトのキーフォーマット

Amazon S3 では、送信先バケットにログオブジェクトをアップロードする際に、次のオブジェクトキー形式を使用します。

- 日付ベース以外のパーティション分割 – これはオリジナルのログオブジェクトキーの形式です。この形式を選択すると、ログファイルのキー形式は次のとおりになります。

  ```
  [DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
  ```

- 日付ベースのパーティション分割 – 日付ベースのパーティショニングを選択した場合、ログ形式で使用される日付ソースとして、ログファイルのイベント時間または配信時間を選択できます。この形式を使用すると、ログのクエリが簡単になります。

  日付ベースのパーティション分割を選択すると、ログファイルのキー形式は次のとおりになります。

  ```
  [DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/
  [YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
  ```

ログオブジェクトキーでは、YYYY、MM、DD、hh、mm、ss は、ログファイルを配信した年、月、日、時、分、秒をそれぞれ表す数字になります。これらの日付と時刻は協定世界時 (UTC) です。

ある時点で配信されたログファイルには、その時点より前に書き込まれたレコードが含まれます。特定の期間のすべてのログレコードが配信されたかどうかを知る方法はありません。

キーの UniqueString コンポーネントは、ファイルの上書きを防止するためのものです。意味はないため、ログ処理ソフトウェアでは無視されます。

ログを配信する方法

Amazon S3 は定期的にアクセスログレコードを収集し、レコードをまとめてログファイルを作成し、そのログファイルを、ログオブジェクトとして送信先バケットにアップロードします。複数のソースバケットでログ記録の配信先が同じ送信先バケットである場合、これらのすべてのソースバ

ケットのアクセスログが送信先バケットに配信されます。ただし、各ログオブジェクトは、ソースバケット別にアクセスログレコードをレポートします。

Amazon S3 は特別なログ配信アカウントを使用してサーバーアクセスログを書き込みます。このような書き込みは、通常のアクセスコントロールの制約に従います。送信先バケットのバケットポリシーを更新して、アクセスログ配信用のログ記録サービスプリンシパル (`logging.s3.amazonaws.com`) へのアクセス権を付与することをお勧めします。バケットのアクセスコントロールリスト (ACL) を介して S3 ログ配信グループにアクセスログ配信のためのアクセスを付与することもできます。ただし、バケット ACL を使用して S3 ログ配信グループへのアクセスを許可することはお勧めしません。

サーバーアクセスのログ記録を有効にして、バケットポリシーを通じてアクセスログ配信のアクセスを許可する場合は、送信先バケットのバケットポリシーを更新して、`s3:PutObject` にログ記録サービスプリンシパルのアクセスを許可します。Amazon S3 コンソールを使用してサーバーアクセスのログ記録を有効にすると、コンソールは、送信先バケットのバケットポリシーを自動的に更新して、ログ記録サービスのプリンシパルにこれらの許可を付与します。サーバーアクセスのログ配信許可の付与の詳細については、ログ配信許可 を参照してください。

> **Note**
>
> S3 は、VPC エンドポイントポリシーで拒否されている場合、または VPC ポリシーが評価される前にリクエストが失敗した場合、VPC エンドポイントリクエストの CloudTrail ログまたはサーバーアクセスログのリクエスタまたはバケット所有者への配信をサポートしません。

S3 オブジェクト所有権のバケット所有者強制設定

送信先バケットが、オブジェクト所有権のバケット所有者強制設定を使用している場合、ACL は無効になり、アクセス許可への影響はなくなります。送信先バケットのバケットポリシーを更新して、アクセスログ配信用のログ記録サービスプリンシパルへのアクセス権を付与する必要があります。オブジェクトの所有権の詳細については、「サーバーアクセスのログ記録用の S3 ログ配信グループへのアクセスを付与する」を参照してください。

ベストエフォート型のサーバーログ配信

サーバーアクセスログレコードの配信は、ベストエフォートベースで行われます。ログ記録用に適切にバケットを設定した場合、そのバケットへのほとんどのリクエストについてログレコードが配信されます。ほとんどのログレコードは、記録された時間から数時間以内に配信されますが、配信間隔は短くなる場合もあります。

サーバーログの完全性や適時性は保証されません。リクエストのログレコードが、リクエストが実際に処理されてからかなり後に配信されたり、配信すらされないこともあり得ます。ログレコードが重複している場合さえあります。サーバーログの目的は、バケットに対するトラフィックの特性を理解することです。ログレコードが損失したり、ログレコードが重複したりすることはまれであるとはいえ、すべてのリクエストが完全に報告されるとは限りません。

サーバーログ作成機能はベストエフォート型であるため、使用状況レポートには、サーバーログに記録されていないアクセスリクエストが含まれる場合があります。このような使用状況レポートは、AWS Billing and Cost Management コンソールの [Cost & usage reports] で確認できます。

バケットのログ記録ステータスの変更が有効になるまでには時間がかかる

バケットのログ記録ステータスの変更がログファイルの配信に反映されるまでには時間がかかります。例えば、バケットのログを有効にする場合、その後数時間に行われるリクエストは記録される場合もあれば、されない場合もあります。ログ記録の送信先バケットをバケット A からバケット B に変更すると、その後 1 時間は一部のログがバケット A に引き続き配信されたり、新しいターゲットバケット B に配信されたりします。いずれにしても、最終的に新しい設定が有効になるため、ユーザー側の操作は一切必要ありません。

ログ記録とログファイルの詳細については、次のセクションを参照してください。

トピック

- Amazon S3 サーバーアクセスログを有効にします。
- Amazon S3 サーバーアクセスログの形式
- Amazon S3 ログファイルの削除
- Amazon S3 アクセスログを使用したリクエストの識別

Amazon S3 サーバーアクセスログを有効にします。

サーバーアクセスのログには、Amazon S3 バケットに対するリクエストの詳細が記録されます。サーバーアクセスのログは、多くのアプリケーションに役立ちます。例えば、アクセスのログ情報は、セキュリティやアクセスの監査に役立ちます。この情報は、顧客基盤の把握や Amazon S3 の請求の理解にもつながります。

デフォルトでは、Amazon S3 によってサーバーアクセスログは収集されません。ログ記録を有効にすると、Amazon S3 は、ソースバケットのアクセスログを選択された送信先バケット(ターゲットバケット) に配信します。ソースと送信先バケットの両方が同じ AWS リージョン にあり、同じ AWS アカウント が所有している必要があります。

アクセスログのレコードには、バケットに対するリクエストの詳細が取り込まれます。この情報には、リクエストタイプ、リクエストで指定したリソース、リクエストを処理した日時などが含まれます。ログ記録の基本の詳細については、サーバーアクセスログによるリクエストのログ記録 を参照してください。

> **Important**
>
> - Amazon S3 バケットに対してサーバーアクセスログ記録を有効にしても追加料金はかかりません。ただし、システムが配信するいずれのログファイルの格納に対しても通常の料金がかかります (ログはいつでも削除できます)。ログファイルの配信にデータ転送料金はかかりません。ただし、ログファイルへのアクセスには通常のデータ転送料金がかかります。
> - 送信先バケットでサーバーアクセスのログ記録が有効になっているべきではありません。ログの保存先のバケットとして、ソースバケットと同じリージョンにあるユーザー所有のバケットを指定できます。これにはソースバケット自体も含まれます。ただし、ソースバケットにログを配信すると、ログの無限ループが発生するため、お勧めしません。ログを管理しやすくするため、アクセスログは別のバケットに保存することをお勧めします。詳細については、「ログ配信を有効にするにはどうすればよいですか?」を参照してください。
> - S3 オブジェクトロックが有効になっている S3 バケットは、サーバーアクセスログの送信先バケットとして使用できません。送信先バケットにはデフォルト保持期間設定を指定できません。
> - 送信先バケットでは [リクエスタ支払い] を有効にすることはできません。
> - 送信先バケットでデフォルトのバケット暗号化を使用できるのは、256 ビットの高度暗号化標準 (AES-256) を使用する Amazon S3 マネージドキー S3 によるサーバー側の暗号化を使用する場合のみです。AWS Key Management Service (AWS KMS) キーを使用したデフォルトのサーバー側の暗号化 (SSE-KMS) はサポートされていません。

Amazon S3 コンソール、Amazon S3 API、AWS Command Line Interface (AWS CLI)、AWS SDK を使用して、サーバーアクセスのログ記録を有効または無効にできます。

ログ配信許可

Amazon S3 は特別なログ配信アカウントを使用してサーバーアクセスログを書き込みます。このような書き込みは、通常のアクセスコントロールの制約に従います。アクセスログを配信するには、ロ

ギングサービスプリンシパル (logging.s3.amazonaws.com) に送信先バケットへのアクセス権を付与する必要があります。

Amazon S3 にログ配信アクセス許可を付与するには、送信先バケットの S3 オブジェクト所有権の設定に応じて、バケットポリシーまたはバケットアクセスコントロールリスト (ACL) を使用できます。ただし、ACL の代わりにバケットポリシーを使用することをお勧めします。

S3 オブジェクト所有権のバケット所有者強制設定

送信先バケットが、オブジェクト所有権のバケット所有者強制設定を使用している場合、ACL は無効になり、アクセス許可への影響はなくなります。この場合は、送信先バケットのバケットポリシーを更新して、アクセスログ配信用のログ記録サービスプリンシパルへのアクセス権を付与する必要があります。S3 ログ配信グループへのアクセスを許可すると、バケット ACL を更新できません。また、送信先権限 (別名ターゲット権限) PutBucketLogging を設定に含めることもできません。

アクセスログ配信用の既存のバケット ACL をバケットポリシーに移行する方法については、「サーバーアクセスのログ記録用の S3 ログ配信グループへのアクセスを付与する」を参照してください。オブジェクトの所有権の詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。新しいバケットを作成するとき、ACL はデフォルトでは無効になっています。

バケットポリシーを使用したアクセスの付与

送信先バケットでバケットポリシーを使用してアクセスを許可するには、バケットポリシーを更新して、ログ記録サービスプリンシパルに s3:PutObject アクセス許可を付与します。Amazon S3 コンソールを使用してサーバーアクセスのログ記録を有効にすると、コンソールは、送信先バケットのバケットポリシーを自動的に更新して、ログ記録サービスプリンシパルにこのようなアクセス許可を付与します。プログラムでサーバーアクセスのログ記録を有効にする場合、送信先バケットのバケットポリシーを手動で更新して、アクセスログ配信用のログ記録サービスプリンシパルへのアクセス権を付与する必要があります。

ログ記録サービスプリンシパルのリソースへのアクセスを制限するバケットポリシーの例については、「the section called “バケットポリシーを使用して、ロギングサービスプリンシパルに許可を付与”」を参照してください。

バケット ACL を使用したアクセスの付与

代わりに、バケット ACL を使用して、アクセスログ配信のアクセスを許可できます。S3 ログ配信グループに WRITE と READ_ACP の許可を付与する許可エントリーをバケット ACL に追加します。ただし、バケット ACL を使用して S3 ログ配信グループへのアクセスを許可することはお勧めしません。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照して

ください。アクセスログ配信用の既存のバケット ACL をバケットポリシーに移行する方法については、「サーバーアクセスのログ記録用の S3 ログ配信グループへのアクセスを付与する」を参照してください。ログ記録サービスプリンシパルのリソースへのアクセスを制限する ACL の例については、「the section called “バケット ACL を使用して、ログ配信グループにアクセス許可を付与します。”」を参照してください。

バケットポリシーを使用して、ロギングサービスプリンシパルに許可を付与

このバケットポリシーの例は、s3:PutObject アクセス許可をログ記録サービスプリンシパル (logging.s3.amazonaws.com) に付与します。このバケットポリシーを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。次のポリシーでは、*example-s3-destination-bucket* はサーバーアクセスログが配信される送信先バケットで、*example-s3-source-bucket* は送信元バケットです。*EXAMPLE-LOGGING-PREFIX* は、ログオブジェクトに使用するオプションの送信先のプレフィックス (別称 ターゲットプレフィックス) です。*SOURCE-ACCOUNT-ID* は、AWS アカウント ソースバケットを所有するグループです。

> **Note**
>
> バケットポリシーに Deny ステートメントがある場合は、それらが Amazon S3 からのアクセスログの配信を防止していないことを確認します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::example-s3-destination-bucket/EXAMPLE-LOGGING-
PREFIX*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::example-s3-source-bucket"
                },
                "StringEquals": {
```

```
                    "aws:SourceAccount": "SOURCE-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

バケット ACL を使用して、ログ配信グループにアクセス許可を付与します。

> **Note**
>
> セキュリティ上のベストプラクティスとして、Amazon S3 はすべての新しいバケットのアクセスコントロールリスト (ACL) をデフォルトで無効にします。Amazon S3 コンソールでの ACL アクセス権限の詳細については、「ACL の設定」を参照してください。

バケット ACL を使用してログ配信グループに許可を付与することはできますが、この方法はお勧めしません。ただし、送信先バケットが、オブジェクト所有権のバケット所有者強制設定を使用している場合、バケット ACL や オブジェクト ACL を設定することはできません。また、送信先権限 (別名ターゲット権限) PutBucketLogging を設定に含めることもできません。代わりに、ロギングサービスプリンシパル (logging.s3.amazonaws.com) にアクセス許可を付与するために、バケットポリシーを使用する必要があります。詳細については、「ログ配信許可」を参照してください。

バケット ACL では、ログ配信グループは次の URL で示されます。

```
http://acs.amazonaws.com/groups/s3/LogDelivery
```

WRITE と READ_ACP (ACL 読み取り) アクセス許可を付与するには、送信先バケット ACL に次の権限を追加します。

```
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>WRITE</Permission>
</Grant>
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>READ_ACP</Permission>
```

```
</Grant>
```

プログラムを使用して ACL アクセス権限を追加する例については、ACL の設定 を参照してください。

> **Important**
>
> バケットで AWS CloudFormation 使用して Amazon S3 サーバーアクセスロギングを有効にし、ACL を使用して S3 ログ配信グループへのアクセスを許可する場合は、CloudFormation テンプレートにも「`AccessControl": "LogDeliveryWrite"`」を追加する必要があります。このようなアクセス許可はバケット ACL を作成することによってのみ付与できます。CloudFormation ではバケットのカスタム ACL を作成できないため、これを実行することは重要です。CloudFormation で使用できるのは既定 ACL のみです。

サーバーアクセスのログ記録を有効にします

Amazon S3 コンソール、Amazon S3 REST API、AWS SDK、AWS CLI を使用してサーバーアクセスのログ記録を有効にするには、次の手順を実行します。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、サーバーアクセスログ記録を有効にするバケットの名前を選択します。
3. [プロパティ] を選択します。
4. [サーバーアクセスのログ記録] で、[編集] を選択します。
5. [Server access logging] (サーバーアクセスのログ記録) で[Enable] (有効) を選択します。
6. [送信先バケット] の下で、バケットとオプションのプレフィックスを指定します。プレフィックスを指定する場合は、ログを検出しやすくするために、プレフィックスの後にスラッシュ (/) を付けることをお勧めします。

   > **Note**
   >
   > スラッシュ (/) を使用したプレフィックスを指定すると、ログオブジェクトが見つけやすくなります。例えば、プレフィックスの値として `logs/` を指定すると、次のとおり Amazon S3 で作成する各ログオブジェクトのキーの先頭に `logs/` というプレフィックスが付けられます。

```
logs/2013-11-01-21-32-16-E568B2907131C0C0
```

プレフィックス値 logs を指定すると、ログオブジェクトは次のように表示されます。

```
logs2013-11-01-21-32-16-E568B2907131C0C0
```

7. [ログオブジェクトキーの形式] の下で、次を設定します。

   - 日付ベース以外のパーティション分割を選択するには、[DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString] を選択します。
   - 日付ベースでパーティション分割するには、[DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString] を選択してから、[S3 イベント時刻] または [ログファイルの配信時刻] を選択します。

8. [Save changes] (変更の保存) をクリックします。

   バケットでサーバーアクセスのログ記録を有効にすると、コンソールはソースバケットでのログ記録を有効にして、送信先バケットのバケットポリシーを更新し、ログ記録サービスプリンシパル (logging.s3.amazonaws.com) に s3:PutObject アクセス許可を付与します。バケットポリシーの詳細については、バケットポリシーを使用して、ロギングサービスプリンシパルに許可を付与 を参照してください。

   ログは 送信先バケットで確認できます。サーバーアクセスのログ記録を有効にすると、ログがターゲットバケットに配信されるまでに数時間かかることがあります。ログ配信の方法と間隔の詳細については、ログを配信する方法 を参照してください。

詳細については、「S3 バケットのプロパティを表示するには」を参照してください。

REST API の使用

ログ記録を有効にするには、PutBucketLogging リクエストを送信してソースバケットにログ記録の設定を追加します。リクエストでは、送信先バケット (別名 ターゲットバケット) を指定し、必要に応じて、すべてのログオブジェクトキーに使用するプレフィックスを指定します。

送信先バケットとして *example-s3-destination-bucket* を、プレフィックスとして *logs/* を指定する例は次のとおりです。

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>example-s3-destination-bucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
```

送信先バケットとして *example-s3-destination-bucket* を、プレフィックスとして *logs/* を、ログオブジェクトキーの形式として EventTime を指定する例は次のとおりです。

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>example-s3-destination-bucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
    <TargetObjectKeyFormat>
      <PartitionedPrefix>
         <PartitionDateSource>EventTime</PartitionDateSource>
      </PartitionedPrefix>
  </TargetObjectKeyFormat>
  </LoggingEnabled>
</BucketLoggingStatus>
```

ログオブジェクトはログ配信アカウントが作成および所有し、バケット所有者はログオブジェクトへの完全な許可を付与されます。必要に応じて、送信先権限 (別名ターゲット権限) をして、他のユーザーがログにアクセスできるようにアクセス許可を付与できます。詳細については、「PutBucketLogging」を参照してください。

> **Note**
>
> 送信先バケットが、オブジェクト所有権のバケット所有者強制設定を使用している場合、送信先権限を使用してその他のユーザーにアクセス許可を付与することはできません。その他のユーザーにアクセス許可を付与するには、送信先バケットのバケットポリシーを更新します。詳細については、「ログ配信許可」を参照してください。

バケットのログ記録設定を取得するには、GetBucketLogging API オペレーションを使用します。

ログ記録設定を削除するには、PutBucketLogging リクエストで空の BucketLoggingStatus を指定して送信します。

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
</BucketLoggingStatus>
```

バケットでログ記録を有効にするには、Amazon S3 API または AWS SDK ラッパーライブラリを使用できます。

AWS SDK の使用

バケットでログ記録を有効化する例は次のとおりです。ソースバケットと送信先 (ターゲット) バケットの 2 つのバケットを作成する必要があります。 この例ではまず、送信先バケットのバケット ACL を更新します。この例では、まず送信先バケットにログを書き込むために必要なアクセス許可をログ配信グループに付与して、次にソースバケットでのログ記録を有効にしています。

このような例では、オブジェクト所有権のバケット所有者の強制設定を使用する送信先バケットでは機能しません。

送信先 (ターゲット) バケットが、オブジェクト所有権にバケット所有者の強制設定を使用している場合、バケット ACL や オブジェクト ACL を設定することはできません。PutBucketLogging 設定には、送信先 (ターゲット) 権限を含めることもできません。ロギングサービスプリンシパル (logging.s3.amazonaws.com) に許可を付与するために、バケットポリシーを使用する必要があります。詳細については、「ログ配信許可」を参照してください。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
```

```
    /// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
    /// this example. The first is the bucket for which you wish to enable
    /// logging, and the second is the location where you want to store the
    /// logs.
    /// </summary>
    public class ServerAccessLogging
    {
        private static IConfiguration _configuration = null!;

        public static async Task Main()
        {
            LoadConfig();

            string bucketName = _configuration["BucketName"];
            string logBucketName = _configuration["LogBucketName"];
            string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
            string accountId = _configuration["AccountId"];

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Update bucket policy for target bucket to allow delivery of
logs to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
```

```
                Console.WriteLine($"Error: {e.Message}");
            }
        }

        /// <summary>
        /// This method grants appropriate permissions for logging to the
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" +
logPrefix + @"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3:::" + sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
```

```
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
 enable logging:");
            Console.WriteLine(newPolicy);

            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be
 stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
 used
        /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
 you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
 logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
```

```
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load
  local settings.
                .Build();
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「PutBucketLogging」を参照してください。

Java

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLoggingStatus;
import software.amazon.awssdk.services.s3.model.LoggingEnabled;
import software.amazon.awssdk.services.s3.model.PartitionedPrefix;
import software.amazon.awssdk.services.s3.model.PutBucketLoggingRequest;
import software.amazon.awssdk.services.s3.model.TargetObjectKeyFormat;

// Class to set a bucket policy on a target S3 bucket and enable server access
 logging on a source S3 bucket.
```

```
public class ServerAccessLogging {
    private static S3Client s3Client;

    public static void main(String[] args) {
        String sourceBucketName = "SOURCE-BUCKET";
        String targetBucketName = "TARGET-BUCKET";
        String sourceAccountId = "123456789012";
        String targetPrefix = "logs/";

        // Create S3 Client.
        s3Client = S3Client.builder().
                region(Region.US_EAST_2)
                .build();

        // Set a bucket policy on the target S3 bucket to enable server access
 logging by granting the
        // logging.s3.amazonaws.com principal permission to use the PutObject
 operation.
        ServerAccessLogging serverAccessLogging = new ServerAccessLogging();
        serverAccessLogging.setTargetBucketPolicy(sourceAccountId, sourceBucketName,
 targetBucketName);

        // Enable server access logging on the source S3 bucket.
        serverAccessLogging.enableServerAccessLogging(sourceBucketName,
 targetBucketName,
                targetPrefix);

    }

    // Function to set a bucket policy on the target S3 bucket to enable server
 access logging by granting the
    // logging.s3.amazonaws.com principal permission to use the PutObject operation.
    public void setTargetBucketPolicy(String sourceAccountId, String
 sourceBucketName, String targetBucketName) {
        String policy = "{\n" +
                "    \"Version\": \"2012-10-17\",\n" +
                "    \"Statement\": [\n" +
                "        {\n" +
                "            \"Sid\": \"S3ServerAccessLogsPolicy\",\n" +
                "            \"Effect\": \"Allow\",\n" +
                "            \"Principal\": {\"Service\": \"logging.s3.amazonaws.com
\"},\n" +
                "            \"Action\": [\n" +
                "                \"s3:PutObject\"\n" +
```

```
                "            ],\n" +
                "            \"Resource\": \"arn:aws:s3:::" + targetBucketName + "/*
\",\n" +
                "            \"Condition\": {\n" +
                "                \"ArnLike\": {\n" +
                "                    \"aws:SourceArn\": \"arn:aws:s3:::" +
 sourceBucketName + "\"\n" +
                "                },\n" +
                "                \"StringEquals\": {\n" +
                "                    \"aws:SourceAccount\": \"" + sourceAccountId +
 "\"\n" +
                "                }\n" +
                "            }\n" +
                "        }\n" +
                "    ]\n" +
                "}";
        s3Client.putBucketPolicy(b -> b.bucket(targetBucketName).policy(policy));
    }

    // Function to enable server access logging on the source S3 bucket.
    public void enableServerAccessLogging(String sourceBucketName, String
 targetBucketName,
            String targetPrefix) {
        TargetObjectKeyFormat targetObjectKeyFormat =
 TargetObjectKeyFormat.builder()

 .partitionedPrefix(PartitionedPrefix.builder().partitionDateSource("EventTime").build())
                .build();
        LoggingEnabled loggingEnabled = LoggingEnabled.builder()
                .targetBucket(targetBucketName)
                .targetPrefix(targetPrefix)
                .targetObjectKeyFormat(targetObjectKeyFormat)
                .build();
        BucketLoggingStatus bucketLoggingStatus = BucketLoggingStatus.builder()
                .loggingEnabled(loggingEnabled)
                .build();
        s3Client.putBucketLogging(PutBucketLoggingRequest.builder()
                .bucket(sourceBucketName)
                .bucketLoggingStatus(bucketLoggingStatus)
                .build());
    }

}
```

AWS CLI の使用

S3 バケットがある各 AWS リージョンに専用のログ記録バケットを作成することをお勧めします。その後、Amazon S3 アクセスログをその S3 バケットに配信します。詳細と例については、「AWS CLI リファレンス」の「put-bucket-logging」を参照してください。

送信先 (ターゲット) バケットが、オブジェクト所有権にバケット所有者の強制設定を使用している場合、バケット ACL や オブジェクト ACL を設定することはできません。PutBucketLogging 設定には、送信先 (ターゲット) 権限を含めることもできません。ロギングサービスプリンシパル (logging.s3.amazonaws.com) に許可を付与するために、バケットポリシーを使用する必要があります。詳細については、「ログ配信許可」を参照してください。

Example – 2 つのリージョンの 5 つのバケットでアクセスログを有効にする

この例では、次の 5 つのバケットがあります。

- 1-DOC-EXAMPLE-BUCKET1-us-east-1
- 2-DOC-EXAMPLE-BUCKET1-us-east-1
- 3-DOC-EXAMPLE-BUCKET1-us-east-1
- 1-DOC-EXAMPLE-BUCKET1-us-west-2
- 2-DOC-EXAMPLE-BUCKET1-us-west-2

> **Note**
>
> 次の手順の最後のステップでは、ログ記録バケットを作成して、そのバケットでサーバーアクセスのログ記録を有効にするために使用できる bash スクリプトの例が示されています。このようなスクリプトを使用するには、次の手順で説明されるとおり、policy.json と logging.json ファイルを作成する必要があります。

1. 米国西部 (オレゴン) リージョンと米国東部 (バージニア北部) リージョンに 2 つのログ記録先バケットを作成して、次の名前を付けます。
   - DOC-EXAMPLE-BUCKET1-logs-us-east-1
   - DOC-EXAMPLE-BUCKET1-logs-us-west-2
2. このステップの後半では、次のとおりサーバーアクセスのログ記録を有効にします。

- 1-DOC-EXAMPLE-BUCKET1-us-east-1 は、プレフィックス DOC-EXAMPLE-BUCKET1-logs-us-east-1 を使用して S3 バケット 1-DOC-EXAMPLE-BUCKET1-us-east-1 にログを記録します。
- 2-DOC-EXAMPLE-BUCKET1-us-east-1 は、プレフィックス DOC-EXAMPLE-BUCKET1-logs-us-east-1 を使用して S3 バケット 2-DOC-EXAMPLE-BUCKET1-us-east-1 にログを記録します。
- 3-DOC-EXAMPLE-BUCKET1-us-east-1 は、プレフィックス DOC-EXAMPLE-BUCKET1-logs-us-east-1 を使用して S3 バケット 3-DOC-EXAMPLE-BUCKET1-us-east-1 にログを記録します。
- 1-DOC-EXAMPLE-BUCKET1-us-west-2 は、プレフィックス DOC-EXAMPLE-BUCKET1-logs-us-west-2 を使用して S3 バケット 1-DOC-EXAMPLE-BUCKET1-us-west-2 にログを記録します。
- 2-DOC-EXAMPLE-BUCKET1-us-west-2 は、プレフィックス DOC-EXAMPLE-BUCKET1-logs-us-west-2 を使用して S3 バケット 2-DOC-EXAMPLE-BUCKET1-us-west-2 にログを記録します。

3. バケット ACL またはバケットポリシーを使用して、サーバーアクセスのログ記録の配信の許可を付与します。

   - バケットポリシーの更新 (推奨) – ロギングサービスプリンシパルにアクセス許可を付与するには、次の put-bucket-policy コマンドを使用します。*example-s3-destination-bucket-logs* を、宛先のバケットの名前に置き換えます。

     ```
     aws s3api put-bucket-policy --bucket example-s3-destination-bucket-logs --policy
      file://policy.json
     ```

     Policy.json は、次のバケットポリシーを含む現在のフォルダ内の JSON ドキュメントです。このバケットポリシーを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。次のポリシーでは、*example-s3-destination-bucket-logs* はサーバーアクセスログが配信される送信先バケットであり、*example-s3-source-bucket* はソースバケットです。*SOURCE-ACCOUNT-ID* は、ソースバケットを所有する AWS アカウント です。

     ```
     {
         "Version": "2012-10-17",
         "Statement": [
     ```

```
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::example-s3-destination-bucket-logs/*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::example-s3-source-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

- バケット ACL の更新 – S3 ログ配信グループに許可を付与するには、次の put-bucket-acl コマンドを使用します。*example-s3-destination-bucket-logs* は、送信先 (ターゲット) バケット名前に置き換えます。

```
aws s3api put-bucket-acl --bucket example-s3-destination-bucket-logs  --grant-
write URI=http://acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp
 URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

4. 次に、(次の 3 つの例のうちのいずれかに基づいた) ログ記録設定を含む logging.json ファイルを作成します。logging.json ファイルを作成したら、次の put-bucket-logging コマンドを使用してログ記録設定を適用できます。*example-s3-destination-bucket-logs* は、送信先 (ターゲット) バケット名前に置き換えます。

```
aws s3api put-bucket-logging --bucket example-s3-destination-bucket-logs --bucket-
logging-status file://logging.json
```

> **Note**
>
> この put-bucket-logging コマンドを使用して各送信先バケットにログ記録設定を適用する代わりに、次のステップで提供される bash スクリプトのいずれかを使用できます。このようなスクリプトを使用するには、次の手順のとおり、policy.json と logging.json ファイルを作成する必要があります。

logging.json ファイルは、ログ記録設定を含む現在のフォルダ内の JSON ドキュメントです。送信先バケットがオブジェクト所有権に対してバケット所有者の強制設定を使用している場合、ログ記録設定に送信先 (ターゲット) の権限を含めることはできません。詳細については、「ログ配信許可」を参照してください。

Example — 送信先 (ターゲット) 権限なしの **logging.json**

次の例の logging.json ファイルには、送信先 (ターゲット) 権限が含まれていません。そのため、オブジェクト所有権のバケット所有者の強制設定を使用する送信先 (ターゲット) バケットにこの設定を適用できます。

```
{
    "LoggingEnabled": {
        "TargetBucket": "example-s3-destination-bucket-logs",
        "TargetPrefix": "example-s3-destination-bucket/"
     }
 }
```

Example — 送信先 (ターゲット) 権限のある **logging.json**

次の例の logging.json ファイルには、送信先 (ターゲット) 権限が含まれています。

送信先バケットが、オブジェクト所有権のバケット所有者の強制設定を使用している場合、PutBucketLogging 設定に送信先 (ターゲット) 権限を含めることはできません。詳細については、「ログ配信許可」を参照してください。

```
{
    "LoggingEnabled": {
```

```
          "TargetBucket": "example-s3-destination-bucket-logs",
          "TargetPrefix": "example-s3-destination-bucket/",
          "TargetGrants": [
               {
                  "Grantee": {
                      "Type": "AmazonCustomerByEmail",
                      "EmailAddress": "user@example.com"
                   },
                  "Permission": "FULL_CONTROL"
               }
           ]
       }
   }
```

Example – ログオブジェクトキーを S3 イベント時刻に設定した **logging.json**

次の logging.json ファイルは、ログオブジェクトキーの形式を S3 イベント時刻に変更します。暗号化キーの使用の詳細については、「the section called “ログ配信を有効にするにはどうすればよいですか?”」を参照してください。

```
  {
    "LoggingEnabled": {
        "TargetBucket": "example-s3-destination-bucket-logs",
        "TargetPrefix": "example-s3-destination-bucket/",
        "TargetObjectKeyFormat": {
            "PartitionedPrefix": {
                "PartitionDateSource": "EventTime"
            }
         }
    }
}
```

5. アカウントのすべてのバケットに対するアクセスログを追加するには、次の bash スクリプトのいずれかを使用します。*example-s3-destination-bucket-logs* は、送信先 (ターゲット) バケットの名前に置き換え、*us-west-2* はバケットが配置されているリージョンの名前に置き換えます。

> **Note**
>
> これはすべてのバケットが同じリージョンにある場合にのみ機能します。複数のリージョンにバケットがある場合は、スクリプトを調整する必要があります。

Example — バケットポリシーによるアクセスを許可し、アカウントのバケットのロギングを追加します

```
  loggingBucket='example-s3-destination-bucket-logs'
  region='us-west-2'


  # Create the logging bucket.
  aws s3 mb s3://$loggingBucket --region $region

  aws s3api put-bucket-policy --bucket $loggingBucket --policy file://policy.json

  # List the buckets in this account.
  buckets="$(aws s3 ls | awk '{print $3}')"

  # Put a bucket logging configuration on each bucket.
  for bucket in $buckets
      do
        # This if statement excludes the logging bucket.
        if [ "$bucket" != "$loggingBucket" ] ; then
            continue;
        fi
        printf '{
          "LoggingEnabled": {
            "TargetBucket": "%s",
            "TargetPrefix": "%s/"
        }
      }' "$loggingBucket" "$bucket"  > logging.json
      aws s3api put-bucket-logging --bucket $bucket --bucket-logging-status file://
logging.json
      echo "$bucket done"
  done

  rm logging.json
```

```
  echo "Complete"
```

Example — バケット ACL でアクセス権を付与し、アカウントのバケットのロギングを追加します

```
  loggingBucket='example-s3-destination-bucket-logs'
  region='us-west-2'


  # Create the logging bucket.
  aws s3 mb s3://$loggingBucket --region $region

  aws s3api put-bucket-acl --bucket $loggingBucket --grant-write URI=http://
acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp URI=http://
acs.amazonaws.com/groups/s3/LogDelivery

  # List the buckets in this account.
  buckets="$(aws s3 ls | awk '{print $3}')"

  # Put a bucket logging configuration on each bucket.
  for bucket in $buckets
      do
        # This if statement excludes the logging bucket.
        if [ "$bucket" != "$loggingBucket" ] ; then
            continue;
        fi
        printf '{
          "LoggingEnabled": {
            "TargetBucket": "%s",
            "TargetPrefix": "%s/"
        }
      }' "$loggingBucket" "$bucket"  > logging.json
      aws s3api put-bucket-logging --bucket $bucket --bucket-logging-status file://
logging.json
      echo "$bucket done"
  done

  rm logging.json

  echo "Complete"
```

サーバーアクセスログ設定の検証

サーバーアクセスロギングを有効にしたら、以下の手順を実行します。

- 送信先バケットにアクセスして、ログファイルが配信されていることを確認します。アクセスログを設定した後、すべてのリクエストが正しく記録され、配信されるまでに 1 時間以上かかる場合があります。Amazon S3 リクエストメトリクスを使用し、これらのメトリックスに Amazon CloudWatch アラームを設定することで、ログ配信を自動的に検証することもできます。詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。
- ログファイルのコンテンツを開いて読み取ることができることを確認します。

サーバアクセスロギングのトラブルシューティング情報については、サーバーのアクセスログ記録のトラブルシューティング を参照してください。

Amazon S3 サーバーアクセスログの形式

サーバーアクセスのログには、Amazon S3 バケットに対するリクエストの詳細が記録されます。サーバーアクセスログは次の目的で使用できます。

- セキュリティ監査とアクセス監査の実施
- 顧客ベースについて学ぶ
- Amazon S3 の請求書を理解する

このセクションでは、Amazon S3 サーバーアクセスログファイルの形式およびその他の詳細について説明します。

サーバーアクセスのログファイルは、一連のログレコードを改行で区切って構成します。各ログレコードは 1 個のリクエストを表し、各フィールドをスペースで区切って構成します。

次に示すのは、5 個のログレコードで構成されるログの例です。

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DOC-EXAMPLE-BUCKET1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 3E57427F3EXAMPLE
 REST.GET.VERSIONING - "GET /DOC-EXAMPLE-BUCKET1?versioning HTTP/1.1" 200 - 113 - 7 -
 "-" "S3Console/0.4" - s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/
XV/VLi31234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader DOC-EXAMPLE-BUCKET1.s3.us-
west-1.amazonaws.com TLSV1.2 arn:aws:s3:us-west-1:123456789012:accesspoint/example-AP
 Yes
```

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DOC-EXAMPLE-BUCKET1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 891CE47D2EXAMPLE
 REST.GET.LOGGING_STATUS - "GET /DOC-EXAMPLE-BUCKET1?logging HTTP/1.1" 200 -
 242 - 11 - "-" "S3Console/0.4" - 9vKBE6vMhrNiWHZmb2L0mXOcqPGzQOI5XLnCtZNPxev+Hf
+7tpT6sxDwDty4LHBUOZJG96N1234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader DOC-
EXAMPLE-BUCKET1.s3.us-west-1.amazonaws.com TLSV1.2 - -
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DOC-EXAMPLE-BUCKET1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be A1206F460EXAMPLE
 REST.GET.BUCKETPOLICY - "GET /DOC-EXAMPLE-BUCKET1?policy HTTP/1.1" 404
 NoSuchBucketPolicy 297 - 38 - "-" "S3Console/0.4" - BNaBsXZQQDbssi6xMBdBU2sLt
+Yf5kZDmeBUP35sFoKa3sLLeMC78iwEIWxs99CRUrbS4n11234= SigV4 ECDHE-RSA-AES128-GCM-SHA256
 AuthHeader DOC-EXAMPLE-BUCKET1.s3.us-west-1.amazonaws.com TLSV1.2 - Yes
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DOC-EXAMPLE-BUCKET1 [06/Feb/2019:00:01:00 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 7B4A0FABBEXAMPLE
 REST.GET.VERSIONING - "GET /DOC-EXAMPLE-BUCKET1?versioning HTTP/1.1" 200 -
 113 - 33 - "-" "S3Console/0.4" - Ke1bUcazaN1jWuUlPJaxF64cQVpUEhoZKEG/hmy/gijN/
I1DeWqDfFvnpybfEseEME/u7ME1234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader DOC-
EXAMPLE-BUCKET1.s3.us-west-1.amazonaws.com TLSV1.2 - -
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DOC-EXAMPLE-BUCKET1 [06/Feb/2019:00:01:57 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DD6CC733AEXAMPLE REST.PUT.OBJECT s3-dg.pdf "PUT /DOC-EXAMPLE-BUCKET1/
s3-dg.pdf HTTP/1.1" 200 - - 4406583 41754 28 "-" "S3Console/0.4" -
 10S62Zv81kBW7BB6SX4XJ48o6kpcl6LPwEoizZQQxJd5qDSCTLX0TgS37kYUBKQW3+bPdrg1234= SigV4
 ECDHE-RSA-AES128-SHA AuthHeader DOC-EXAMPLE-BUCKET1.s3.us-west-1.amazonaws.com TLSV1.2
 - Yes
```

> **Note**
>
> 任意のフィールドを - に設定して、データが不明または使用不可であること、またはフィールドがこのリクエストに適用されなかったことを示すことができます。

トピック

- ログレコードフィールド
- コピーオペレーションの追加ログ記録
- カスタムアクセスログ情報
- 拡張可能なサーバーアクセスログの形式のプログラミングに関する考慮事項

ログレコードフィールド

次のリストは、ログレコードのフィールドについて説明しています。

Bucket Owner

ソースバケット所有者の正規ユーザー ID。正規ユーザー ID は、別の形式の AWS アカウント ID です。正規ユーザー ID の詳細については、AWS 全般のリファレンス の「AWS アカウント IDID」を参照してください。アカウントの正規ユーザー ID を検索する方法については、「AWS アカウントの正規ユーザー ID を検索する」を参照してください。

エントリ例

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

バケット

リクエストの処理ターゲットのバケットの名前。システムで受け取ったリクエストの形式に誤りがあり、バケットを特定できない場合、そのリクエストはサーバーアクセスログに表示されません。

エントリ例

```
DOC-EXAMPLE-BUCKET1
```

時間

リクエストが受信された時間。これらの日付と時刻は協定世界時 (UTC) です。`strftime()` terminology を使用した形式は次のようになります: `[%d/%b/%Y:%H:%M:%S %z]`

エントリ例

```
[06/Feb/2019:00:00:38 +0000]
```

リモート IP

リクエスタの表面上の IP アドレス。中間プロキシやファイアウォールにより、リクエストを作成したマシンの実際の IP アドレスが不明確になる場合があります。

エントリ例

```
192.0.2.3
```

リクエスタ

リクエスタの正規ユーザー ID。認証されていないリクエストの場合は - です。リクエスタが IAM ユーザーであった場合、このフィールドは、リクエスタの IAM ユーザー名と IAM ユーザーが属する AWS アカウントのルートユーザーを返します。この識別子は、アクセスコントロールに使用されるものと同じです。

エントリ例

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

リクエスタが引き受けたロールを使用している場合、このフィールドは、引き受けた IAM ロールを返します。

エントリ例

```
arn:aws:sts::123456789012:assumed-role/roleName/test-role
```

リクエスト ID

各リクエストを一意に識別するために Amazon S3 で生成される文字列。

エントリ例

```
3E57427F33A59F07
```

操作

ここに表示されているオペレーション は、SOAP.*operation*、REST.*HTTP_method*.*resource_type*、WEBSITE.*HTTP_method*.*resourc* または BATCH.DELETE.OBJECT、または ライフサイクルとログ記録 の S3.action.resource_type として宣言されます。

エントリ例

```
REST.PUT.OBJECT
```

キー

リクエストのキー (オブジェクト名) 部分。

エントリ例

```
/photos/2019/08/puppy.jpg
```

Request-URI

HTTP リクエストメッセージの Request-URI の部分。

エントリ例

```
"GET /DOC-EXAMPLE-BUCKET1/photos/2019/08/puppy.jpg?x-foo=bar HTTP/1.1"
```

HTTP ステータス

レスポンスの HTTP ステータスの数値。

エントリ例

```
200
```

エラーコード

Amazon S3 エラーコード、またはエラーが発生しなかった場合は「-」。

エントリ例

```
NoSuchBucket
```

送信バイト数

送信されたレスポンスのバイト数 (HTTP プロトコルオーバーヘッドを除きます)。ゼロの場合は「-」。

エントリ例

```
2662992
```

オブジェクトのサイズ

該当するオブジェクトの合計サイズ。

エントリ例

```
3462992
```

合計時間

サーバーから見た、リクエストの転送中の時間数 (ミリ秒単位)。これは、リクエストが受信されてから、レスポンスの最終バイトが送信されるまでの時間を計測した値です。クライアント側での計測値は、ネットワーク遅延により長くなる場合があります。

エントリ例

```
70
```

Turn-Around Time

Amazon S3 でリクエストの処理に要した時間数 (ミリ秒単位)。これは、リクエストの最終バイトが受信されてから、レスポンスの先頭バイトが送信されるまでの時間を計測した値です。

エントリ例

```
10
```

Referer

HTTP `Referer` ヘッダーの値 (存在する場合)。一般に、HTTP ユーザーエージェント (ブラウザなど) は、このヘッダーをリクエスト作成時のリンクページや埋め込みページの URL に設定します。

エントリ例

```
"http://www.example.com/webservices"
```

User-Agent

HTTP `User-Agent` ヘッダーの値

エントリ例

```
"curl/7.15.1"
```

バージョン ID

リクエストのバージョン ID、または オペレーションが versionId パラメータを取らない場合は「-」。

エントリ例

```
3HL4kqtJvjVBH40Nrjfkd
```

ホスト ID

x-amz-id-2 または Amazon S3 拡張リクエスト ID。

エントリ例

```
s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234=
```

署名バージョン

署名バージョン SigV2 か SigV4 (リクエストの認証に使用)、または - (認証されていないリクエストの場合)。

エントリ例

```
SigV2
```

暗号スイート

HTTPS リクエストまたは HTTP の - に対してネゴシエートされた Secure Sockets Layer (SSL) 暗号。

エントリ例

```
ECDHE-RSA-AES128-GCM-SHA256
```

認証タイプ

使用されるリクエスト認証のタイプ。認証ヘッダーは AuthHeader、クエリ文字列 (署名付き URL) は QueryString、認証されていないリクエストには「-」。

エントリ例

```
AuthHeader
```

ホストヘッダー

Amazon S3 への接続に使用するエンドポイント。

エントリ例

```
s3.us-west-2.amazonaws.com
```

一部の古いリージョンでは、レガシーエンドポイントがサポートされています。これらのエンドポイントは、サーバーアクセスログまたは AWS CloudTrail ログに表示される場合があります。詳細については、「レガシーエンドポイント」を参照してください。Amazon S3 リージョンとエンドポイントの完全なリストについては、Amazon Web Services 全般のリファレンス の「Amazon S3 エンドポイントとクォータ」を参照してください。

TLS のバージョン

クライアントによってネゴシエートされた Transport Layer Security (TLS) バージョン。値は TLSv1.1、TLSv1.2、TLSv1.3、- のいずれかです (TLS を使用しなかった場合)。

エントリ例

```
TLSv1.2
```

アクセスポイント ARN

リクエストのアクセスポイントの Amazon リソースネーム (ARN) です。アクセスポイントの ARN の形式が不正、または使用されていない場合、このフィールドには「-」が含まれます。アクセスポイントの詳細については、「アクセスポイントの使用」を参照してください。ARN の詳細については、「AWS リファレンスガイド」の「Amazon リソースネーム (ARN)」を参照してください。

エントリ例

```
arn:aws:s3:us-east-1:123456789012:accesspoint/example-AP
```

aclRequired

リクエストの承認のためにアクセスコントロールリスト (ACL) が必要かどうかを示す文字列。リクエストに承認用の ACL が必要な場合、文字列は「Yes」です。ACL が必要なかった場合、文字列は「-」です。ACL の詳細については、アクセスコントロールリスト (ACL) の概要 を参照してください。aclRequired フィールドを使用して ACL を無効にする方法の詳細については、オブジェクトの所有権の制御とバケットの ACL の無効化。 を参照してください。

エントリ例

```
Yes
```

コピーオペレーションの追加ログ記録

コピーオペレーションには GET と PUT が含まれます。このため、コピーオペレーションの実行時には 2 つのログレコードが記録されます。前述のセクションでは、コピーオペレーションの PUT 部分に関連するフィールドを説明しています。次のリストでは、コピーオペレーションの GET 部分に関連するフィールドを説明します。

バケット所有者

コピーされたオブジェクトを格納するバケットの正規ユーザー ID。正規ユーザー ID は、別の形式の AWS アカウント ID です。正規ユーザー ID の詳細については、AWS 全般のリファレンスの「AWS アカウント IDID」を参照してください。アカウントの正規ユーザー ID を検索する方法については、「AWS アカウントの正規ユーザー ID を検索する」を参照してください。

エントリ例

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

バケット

コピー対象オブジェクトのオブジェクトを格納するバケットの名前。

エントリ例

```
DOC-EXAMPLE-BUCKET1
```

時間

リクエストが受信された時間。これらの日付と時刻は協定世界時 (UTC) です。strftime() terminology を使用した形式は次のようになります: [%d/%B/%Y:%H:%M:%S %z]

エントリ例

```
[06/Feb/2019:00:00:38 +0000]
```

リモート IP

リクエスタの表面上の IP アドレス。中間プロキシやファイアウォールにより、リクエストを作成したマシンの実際の IP アドレスが不明確になる場合があります。

エントリ例

```
192.0.2.3
```

リクエスタ

リクエスタの正規ユーザー ID。認証されていないリクエストの場合は - です。リクエスタが IAM ユーザーであった場合、このフィールドは、リクエスタの IAM ユーザー名と IAM ユーザーが属する AWS アカウントのルートユーザーを返します。この識別子は、アクセスコントロールに使用されるものと同じです。

エントリ例

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

リクエスタが引き受けたロールを使用している場合、このフィールドは、引き受けた IAM ロールを返します。

エントリ例

```
arn:aws:sts::123456789012:assumed-role/roleName/test-role
```

リクエスト ID

各リクエストを一意に識別するために Amazon S3 で生成される文字列。

エントリ例

```
3E57427F33A59F07
```

操作

ここに表示されているオペレーション
は、SOAP.*operation*、REST.*HTTP_method*.*resource_type*、WEBSITE.*HTTP_method*.*resourc*
または BATCH.DELETE.OBJECT と表示されます。

エントリ例

```
REST.COPY.OBJECT_GET
```

キー

コピー対象オブジェクトのキー (オブジェクト名) 部分。オペレーションがキーパラメータを取らない場合は「-」。

エントリ例

```
/photos/2019/08/puppy.jpg
```

Request-URI

HTTP リクエストメッセージの Request-URI の部分。

エントリ例

```
"GET /DOC-EXAMPLE-BUCKET1/photos/2019/08/puppy.jpg?x-foo=bar"
```

HTTP ステータス

コピーオペレーションの GET 部分の HTTP ステータスの数値。

エントリ例

```
200
```

エラーコード

コピーオペレーションの GET 部分の Amazon S3 エラーコード、またはエラーがない場合は「-」。

エントリ例

```
NoSuchBucket
```

送信バイト数

送信されたレスポンスのバイト数 (HTTP プロトコルオーバーヘッドを除く)。ゼロの場合は「-」。

エントリ例

```
2662992
```

オブジェクトのサイズ

該当するオブジェクトの合計サイズ。

エントリ例

```
3462992
```

合計時間

サーバーから見た、リクエストの転送中の時間数 (ミリ秒単位)。これは、リクエストが受信されてから、レスポンスの最終バイトが送信されるまでの時間を計測した値です。クライアント側での計測値は、ネットワーク遅延により長くなる場合があります。

エントリ例

```
70
```

Turn-Around Time

Amazon S3 でリクエストの処理に要した時間数 (ミリ秒単位)。これは、リクエストの最終バイトが受信されてから、レスポンスの先頭バイトが送信されるまでの時間を計測した値です。

エントリ例

```
10
```

Referer

HTTP Referer ヘッダーの値 (存在する場合)。一般に、HTTP ユーザーエージェント (ブラウザなど) は、このヘッダーをリクエスト作成時のリンクページや埋め込みページの URL に設定します。

エントリ例

```
"http://www.example.com/webservices"
```

User-Agent

HTTP User-Agent ヘッダーの値

エントリ例

```
"curl/7.15.1"
```

バージョン ID

コピー対象オブジェクトのバージョン ID、または x-amz-copy-source ヘッダーでコピー元の一部として versionId パラメータを指定しなかった場合は「-」。

エントリ例

```
3HL4kqtJvjVBH40Nrjfkd
```

ホスト ID

x-amz-id-2 または Amazon S3 拡張リクエスト ID。

エントリ例

```
s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234=
```

署名バージョン

Signature Version、SigV2 か SigV4 (リクエストの認証に使用)、または - (認証されていないリクエストの場合)。

エントリ例

```
SigV4
```

暗号スイート

HTTPS リクエストまたは HTTP の - に対してネゴシエートされた Secure Sockets Layer (SSL) 暗号。

エントリ例

```
ECDHE-RSA-AES128-GCM-SHA256
```

認証タイプ

使用されるリクエスト認証のタイプ。認証ヘッダーは AuthHeader、クエリ文字列 (署名付き URL) は QueryString、認証されていないリクエストには「-」。

エントリ例

```
AuthHeader
```

ホストヘッダー

Amazon S3 への接続に使用するエンドポイント。

エントリ例

```
s3.us-west-2.amazonaws.com
```

一部の古いリージョンでは、レガシーエンドポイントがサポートされています。これらのエンドポイントは、サーバーアクセスログまたは AWS CloudTrail ログに表示される場合があります。詳細については、「レガシーエンドポイント」を参照してください。Amazon S3 リージョンとエンドポイントの完全なリストについては、Amazon Web Services 全般のリファレンス の「Amazon S3 エンドポイントとクォータ」を参照してください。

TLS のバージョン

クライアントによってネゴシエートされた Transport Layer Security (TLS) バージョン。値は TLSv1.1、TLSv1.2、TLSv1.3、- のいずれかです (TLS を使用しなかった場合)。

エントリ例

```
TLSv1.2
```

アクセスポイント ARN

リクエストのアクセスポイントの Amazon リソースネーム (ARN) です。アクセスポイントの ARN の形式が不正、または使用されていない場合、このフィールドには「-」が含まれます。アクセスポイントの詳細については、「アクセスポイントの使用」を参照してください。ARN の詳細については、「AWS リファレンスガイド」の「Amazon リソースネーム (ARN)」を参照してください。

エントリ例

```
arn:aws:s3:us-east-1:123456789012:accesspoint/example-AP
```

aclRequired

リクエストの承認のためにアクセスコントロールリスト (ACL) が必要かどうかを示す文字列。リクエストに承認用の ACL が必要な場合、文字列は「Yes」です。ACL が必要なかった場合、文字列は「-」です。ACL の詳細については、アクセスコントロールリスト (ACL) の概要 を参照してください。aclRequired フィールドを使用して ACL を無効にする方法の詳細については、オブジェクトの所有権の制御とバケットの ACL の無効化。 を参照してください。

エントリ例

```
Yes
```

カスタムアクセスログ情報

リクエストのアクセスログレコードに保存するカスタム情報を含めることができます。これを行うには、リクエストの URL にカスタムクエリ文字列パラメータを追加します。Amazon S3 では、「x-」で始まるクエリ文字列パラメータは無視されますが、これらのパラメータはログレコードの Request-URI フィールドの一部として、リクエストのアクセスログレコードに追加されます。

例えば、GET の "s3.amazonaws.com/DOC-EXAMPLE-BUCKET1/photos/2019/08/puppy.jpg?x-user=johndoe" リクエストは、"s3.amazonaws.com/DOC-EXAMPLE-BUCKET1/photos/2019/08/puppy.jpg" のリクエストと同じように動作します。ただし 、"x-

user=johndoe" 文字列は関連付けられたログレコードの Request-URI フィールドに含まれている点が異なります。この機能は REST インターフェイスでのみ利用できます。

拡張可能なサーバーアクセスログの形式のプログラミングに関する考慮事項

場合によっては、新しいフィールドを各行末に追加することで、アクセスログレコードの形式を拡張することがあります。したがって、サーバーアクセスログを解析するコードは、理解できない可能性のある後続フィールドを処理するよう作成する必要があります。

Amazon S3 ログファイルの削除

Amazon S3 バケットでサーバーアクセスのログ記録が有効になっていると、時間の経過とともに多数のサーバーログオブジェクトが蓄積される場合があります。アクセスログは、作成後アプリケーションで必要な期間が過ぎたら、削除できます。Amazon S3 のライフサイクル設定でルールを指定しておくことで、このようなオブジェクトのライフサイクルが終了すると、自動的にオブジェクトを削除キューに入れることができます。

共有プレフィックスを使用して、S3 バケット内のオブジェクトのサブセットのライフサイクル設定を定義できます。サーバーアクセスのログ記録設定でプレフィックスを指定した場合は、そのプレフィックスが付いたログオブジェクトを削除するライフサイクル設定ルールを指定できます。

例えば、ログオブジェクトに logs/ というプレフィックスがあるとします。ライフサイクル設定ルールを指定して、指定した期間後に logs/ プレフィックスを持つバケット内のすべてのオブジェクトを削除できます。

ライフサイクル設定についての詳細は、ストレージのライフサイクルの管理 を参照してください。

サーバーアクセスログ記録の一般情報については、サーバーアクセスログによるリクエストのログ記録 を参照してください。

Amazon S3 アクセスログを使用したリクエストの識別

Amazon S3 アクセスログを使用して、Amazon S3 リクエストを識別できます。

> **Note**
>
> - Amazon S3 リクエストを識別するには、Amazon S3 アクセスログの代わりに AWS CloudTrail データイベントを使用することをお勧めします。CloudTrail データイベントは、設定が容易で、より多くの情報が含まれています。詳細については、「CloudTrail を使用した Amazon S3 リクエストの識別」を参照してください。

- 取得したアクセスリクエスト数に応じて、ログの分析にリソースや時間がさらに必要になる場合があります。

トピック

- Amazon Athena を使用したリクエストのアクセスログのクエリ
- Amazon S3 アクセスログを使用した Signature Version 2 リクエストの識別
- Amazon S3 アクセスログを使用したオブジェクトアクセスリクエストの識別

Amazon Athena を使用したリクエストのアクセスログのクエリ

Amazon Athena を使って、Amazon S3 のアクセスログで、Amazon S3 リクエストを識別できます。

Amazon S3 は、サーバーのアクセスログを S3 バケット内のオブジェクトとして保存します。多くの場合、Amazon S3 のログを分析できるツールを使用する方が簡単です。Athena は S3 オブジェクトの分析をサポートしてしるため、Amazon S3 アクセスログに対してクエリを実行するのに使用できます。

Example

次の例は、Amazon Athena で Amazon S3 サーバーアクセスログをクエリする方法を示しています。次の例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

> **Note**
>
> Athena クエリで Amazon S3 のロケーションを指定するには、ログの送信先となるバケットの S3 URI を指定する必要があります。この URI には、次の形式でバケット名とプレフィックスを含める必要があります。`s3://`*example-s3-bucket1*`-logs/`*prefix*`/`

1. https://console.aws.amazon.com/athena/ で Athena コンソールを開きます。
2. クエリエディタで、次のようなコマンドを実行します。*s3_access_logs_db* は、データベースにつける名前に置き換えます。

```
CREATE DATABASE s3_access_logs_db
```

> **Note**
>
> ベストプラクティスとして、データベースは、S3 バケットと同じ AWS リージョンで作成することをお勧めします。

3. クエリエディタで次のようなコマンドを実行して、ステップ 2 で作成したデータベースでテーブルスキーマを作成します。*s3_access_logs_db.mybucket_logs* は、テーブルにつける名前に置き換えます。STRING および BIGINT データ型の値はアクセスログのプロパティです。これらのプロパティは Athena でクエリできます。LOCATION の場合は、前述のように S3 バケットとプレフィックスパスを入力します。

```
CREATE EXTERNAL TABLE `s3_access_logs_db.mybucket_logs`(
  `bucketowner` STRING,
  `bucket_name` STRING,
  `requestdatetime` STRING,
  `remoteip` STRING,
  `requester` STRING,
  `requestid` STRING,
  `operation` STRING,
  `key` STRING,
  `request_uri` STRING,
  `httpstatus` STRING,
  `errorcode` STRING,
  `bytessent` BIGINT,
  `objectsize` BIGINT,
  `totaltime` STRING,
  `turnaroundtime` STRING,
  `referrer` STRING,
  `useragent` STRING,
  `versionid` STRING,
  `hostid` STRING,
  `sigv` STRING,
  `ciphersuite` STRING,
  `authtype` STRING,
  `endpoint` STRING,
  `tlsversion` STRING,
  `accesspointarn` STRING,
  `aclrequired` STRING)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
```

```
  'input.regex'='([^ ]*) ([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\"|-) (-|[0-9]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 (\"[^\"]*\"|-) ([^ ]*)(?: ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*))?.*$')
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET1-logs/prefix/'
```

4. ナビゲーションペインにある、[データベース] で、データベースを選択します。
5. [テーブル] で、テーブル名の横にある、[Preview table (テーブルのプレビュー)] を選択します。

   [結果] ペインに、サーバーアクセスログのデータ (bucketowner、bucket、requestdatetime など) が表示されます。これは、Athena テーブルが正常に作成されたことを意味します。これで Amazon S3 サーバーアクセスログのクエリを実行できるようになりました。

Example – 誰がいつオブジェクトを削除したか (タイムスタンプ、IP アドレス、および IAM ユーザー) を表示する

```
SELECT requestdatetime, remoteip, requester, key
FROM s3_access_logs_db.mybucket_logs
WHERE key = 'images/picture.jpg' AND operation like '%DELETE%';
```

Example – IAM ユーザーによって実行されたすべてのオペレーションを表示する

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
WHERE requester='arn:aws:iam::123456789123:user/user_name';
```

Example – 特定の期間にオブジェクトに対して実行されたすべてのオペレーションを表示する

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
```

```
WHERE Key='prefix/images/picture.jpg'
    AND parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
    BETWEEN parse_datetime('2017-02-18:07:00:00','yyyy-MM-dd:HH:mm:ss')
    AND parse_datetime('2017-02-18:08:00:00','yyyy-MM-dd:HH:mm:ss');
```

Example – 特定の期間、特定の IP アドレスに送信されたデータの量を表示する

```
SELECT coalesce(SUM(bytessent), 0) AS bytessenttotal
FROM s3_access_logs_db.mybucket_logs
WHERE remoteip='192.0.2.1'
AND parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2022-06-01','yyyy-MM-dd')
AND parse_datetime('2022-07-01','yyyy-MM-dd');
```

> **Note**
>
> ログを保持する時間を短縮するために、サーバーアクセスログバケットの Amazon S3 ライフサイクル設定を作成できます。ライフサイクル設定ルールを作成して、定期的にログファイルを削除します。これにより、各クエリで Athena が分析するデータの量が減ります。詳細については、「バケットにライフサイクル設定を設定する」を参照してください。

Amazon S3 アクセスログを使用した Signature Version 2 リクエストの識別

Signature Version 2 の Amazon S3 サポートがオフになります (非推奨)。その後、Amazon S3 は署名バージョン 2 を使用するリクエストを受け入れず、すべてのリクエストに Signature Version 4 署名を使用する必要があります。Amazon S3 アクセスログを使用して、Signature Version 2 アクセスリクエストを識別できます。

> **Note**
>
> Signature Version 2 リクエストを識別するには、Amazon S3 アクセスログの代わりに AWS CloudTrail データイベントを使用することをお勧めします。CloudTrail データイベントは、サーバーアクセスログと比べ、設定が容易で、より多くの情報が含まれています。詳細については、「CloudTrail を使用した Amazon S3 Signature Version 2 リクエストの識別」を参照してください。

Example – 署名バージョン 2 トラフィックを送信しているすべてのリクエスタを表示する

```
SELECT requester, sigv, Count(sigv) as sigcount
FROM s3_access_logs_db.mybucket_logs
GROUP BY requester, sigv;
```

Amazon S3 アクセスログを使用したオブジェクトアクセスリクエストの識別

Amazon S3 サーバーアクセスログに対するクエリを使用して、GET、PUT、DELETE などのオペレーションに対する Amazon S3 オブジェクトアクセスリクエストを識別し、それらのリクエストに関する詳細情報を確認することができます。

次の Amazon Athena クエリの例は、サーバーアクセスログから Amazon S3 に対するすべての PUT オブジェクトリクエストを取得する方法を示しています。

Example – 一定期間内に **PUT** オブジェクトリクエストを送信しているすべてのリクエスタを表示する

```
SELECT bucket_name, requester, remoteip, key, httpstatus, errorcode, requestdatetime
FROM s3_access_logs_db
WHERE operation='REST.PUT.OBJECT' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

次の Amazon Athena クエリの例は、サーバーアクセスログから Amazon S3 に対するすべての GET オブジェクトリクエストを取得する方法を示しています。

Example – 一定期間内に **GET** オブジェクトリクエストを送信しているすべてのリクエスタを表示する

```
SELECT bucket_name, requester, remoteip, key, httpstatus, errorcode, requestdatetime
FROM s3_access_logs_db
WHERE operation='REST.GET.OBJECT' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

次の Amazon Athena のクエリの例は、S3 バケットへのすべての匿名リクエストをサーバーアクセスログから取得する方法を示しています。

Example – 特定の期間にバケットにリクエストを行っているすべての匿名リクエスタを表示する

```
SELECT bucket_name, requester, remoteip, key, httpstatus, errorcode, requestdatetime
FROM s3_access_logs_db.mybucket_logs
WHERE requester IS NULL AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

次の Amazon Athena のクエリは、承認のためにアクセスコントロールリスト (ACL) が必要な S3 バケットへのすべてのリクエストを特定する方法を示しています。この情報を使用して、これらの ACL 権限を適切なバケットポリシーに移行し、ACL を無効にすることができます。これらのバケットポリシーを作成したら、これらのバケットの ACL を無効にできます。ACL の無効化の詳細については、「ACL を無効にする前提条件」を参照してください。

Example — 承認に ACL が必要なリクエストをすべて特定する

```
SELECT bucket_name, requester, key, operation, aclrequired, requestdatetime
FROM s3_access_logs_db
WHERE aclrequired = 'Yes' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2022-05-10:00:00:00','yyyy-MM-dd:HH:mm:ss')
AND parse_datetime('2022-08-10:00:00:00','yyyy-MM-dd:HH:mm:ss')
```

Note

- ニーズに合わせられるように、必要に応じてデータ範囲を変更することができます。
- このクエリの例は、セキュリティのモニタリングにも役立つ場合があります。予期しないまたは不正な IP アドレスやリクエスタからの PutObject または GetObject コールの結果を確認し、バケットへの匿名リクエストを特定できます。
- このクエリでは、ログ記録が有効になった時間以降の情報のみ取得されます。

- AWS CloudTrail ログを使用している場合は、「CloudTrail を使用した S3 オブジェクトへのアクセスの識別」を参照してください。

Amazon CloudWatch によるメトリクスのモニタリング

Amazon S3 の Amazon CloudWatch メトリクスは、Amazon S3 を使用するアプリケーションのパフォーマンスを理解して向上させるのに役立つことがあります。Amazon S3 で CloudWatch を使用する方法は複数あります。

バケットの日次ストレージメトリクス

バケットストレージは、CloudWatch を使用してモニタリングできます。これは、Amazon S3 からのストレージデータを収集し、読み取り可能な日次のメトリクスに加工します。これらの Amazon S3 のストレージメトリクスは 1 日に 1 回報告され、すべてのお客様に追加料金なしで提供されます。

リクエストメトリクス

Amazon S3 リクエストをモニタリングし、オペレーションの問題をすばやく特定して対応します。メトリクスは、処理のレイテンシーの後に 1 分間隔で使用できます。これらの CloudWatch メトリクスは、Amazon CloudWatch カスタムメトリクスと同じレートで請求されます。CloudWatch の料金の詳細については、Amazon CloudWatch の料金をご覧ください。これらのメトリクスの取得をオプトインする方法の詳細については、CloudWatch メトリクスの設定 を参照してください。

有効にすると、すべてのオブジェクトオペレーションのリクエストメトリクスが報告されます。デフォルトでは、Amazon S3 バケットレベルでこれらの 1 分のメトリクスが利用可能です。共有プレフィックス、オブジェクトタグ、またはアクセスポイントを使用してメトリクスにフィルタを定義することもできます。

- アクセスポイント – アクセスポイントは、バケットに接続され、S3 の共有データセットの大規模なデータアクセスの管理を簡素化する名前付きネットワークエンドポイントです。アクセスポイントフィルタを使用すると、アクセスポイントの使用状況に関するインサイトを得ることができます。アクセスポイントの詳細については、アクセスポイントのモニタリングとログ記録 を参照してください。
- プレフィックス – Amazon S3 データモデルはフラット構造ですが、プレフィックスを使用して階層を推測できます。プレフィックスは、バケット内の類似オブジェクトをグループ化できるディレクトリ名に似ています。S3 コンソールはフォルダの概念でこれらのプレフィックスを

サポートします。プレフィックスでフィルタリングした場合、同じプレフィックスを持つオブジェクトがメトリクス設定に含まれます。プレフィックスの詳細については、プレフィックスを使用してオブジェクトを整理する を参照してください。

- タグ – タグは、オブジェクトに追加できるキーと値の名前のペアです。タグを使用すると、オブジェクトを簡単に検索および整理できます。メトリクス設定のフィルタとしてタグを使用して、それらのタグを持つオブジェクトのみがメトリクス設定に含められるようにすることもできます。オブジェクトタグの詳細については、タグを使用してストレージを分類するを参照してください。

共有プレフィックス、オブジェクトタグ、またはアクセスポイントでフィルタリングすると、こうしたメトリクスを特定のビジネスアプリケーション、ワークフロー、または内部組織に合わせることができます。

レプリケーションメトリクス

レプリケーションメトリクス – レプリケーションを保留している S3 API オペレーションの合計数、レプリケーションを保留しているオブジェクトの合計サイズ、およびレプリケート先 AWS リージョン への最大レプリケーション時間、およびレプリケーションに失敗したオペレーションの合計数をモニタリングします。S3 Replication Time Control (S3 RTC) または S3 レプリケーションメトリクスが有効になっているレプリケーションルールがレプリケーションメトリクスを発行します。

詳細については、レプリケーションメトリクスと S3 イベント通知による、進捗状況のモニタリング または S3 Replication Time Control (S3 RTC) を使用してコンプライアンス要件を満たす を参照してください。

Amazon S3 ストレージレンズメトリクス

S3 StorageLens の使用状況とアクティビティのメトリクスを Amazon CloudWatch に公開して、CloudWatch ダッシュボード で運用状態の統一されたビューを作成できます。`AWS/S3/Storage-Lens` 名前空間で S3 Storage Lens のメトリクスは可能です。CloudWatch 公開オプションは、アドバンストメトリクスとレコメンデーションにアップグレードされた S3 StorageLens ダッシュボードで利用できます。S3 Storage Lens の新規または既存のダッシュボード設定に対して CloudWatch 公開オプションを有効にできます。

詳細については、「CloudWatch で S3 Storage Lens のメトリクスをモニタリング」を参照してください。

すべての CloudWatch 統計は 15 か月間保持されるため、履歴情報にアクセスしてウェブアプリケーションまたはサービスの動作をより的確に把握することができます。CloudWatch の詳細につ

いては、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch とは」を参照してください。ユースケースによっては、CloudWatch アラームに追加の設定が必要になる場合があります。例えば、メトリクスの数式を使用してアラームを作成できます。詳細については、「Amazon CloudWatch ユーザーガイド」の「CloudWatch メトリクスを使用する」、「Metric Math を使用する」、「Amazon CloudWatch アラームの使用」、および「メトリクス数式に基づく CloudWatch アラームの作成」を参照してください。

ベストエフォート型の CloudWatch メトリクス配信

CloudWatch メトリクスは、ベストエフォートで配信されます。リクエストメトリクスを持つ Amazon S3 オブジェクトのほとんどのリクエストにより、データポイントが CloudWatch に送信されます。

メトリクスの完全性や適時性は保証されません。特定のリクエストのデータポイントが、リクエストが実際に処理されたよりも後のタイムスタンプで返される場合があります。1 分のデータポイントは CloudWatch を通じて利用可能になるまで遅延する場合や、まったく提供されない場合があります。CloudWatch リクエストメトリクスは、バケットに対するトラフィックの特性をほぼリアルタイムで示します。すべてのリクエストを完全に報告するためのものではありません。

この機能はベストエフォート型であるため、請求情報とコスト管理ダッシュボードで利用できるレポートには、バケットメトリクスに表示されない 1 つ以上のアクセスリクエストが含まれることがあります。

詳細については、以下のトピックを参照してください。

トピック

- メトリクスとディメンション
- CloudWatch メトリクスへのアクセス
- CloudWatch メトリクスの設定

メトリクスとディメンション

Amazon S3 が Amazon CloudWatch に送信するストレージメトリクスとディメンションを次の表に一覧表示します。

ベストエフォート型の CloudWatch メトリクス配信

CloudWatch メトリクスは、ベストエフォートで配信されます。リクエストメトリクスを持つ Amazon S3 オブジェクトのほとんどのリクエストにより、データポイントが CloudWatch に送信されます。

メトリクスの完全性や適時性は保証されません。特定のリクエストのデータポイントが、リクエストが実際に処理されたよりも後のタイムスタンプで返される場合があります。1 分のデータポイントは CloudWatch を通じて利用可能になるまで遅延する場合や、まったく提供されない場合があります。CloudWatch リクエストメトリクスは、バケットに対するトラフィックの特性をほぼリアルタイムで示します。すべてのリクエストを完全に報告するためのものではありません。

この機能はベストエフォート型であるため、請求情報とコスト管理ダッシュボードで利用できるレポートには、バケットメトリクスに表示されない 1 つ以上のアクセスリクエストが含まれることがあります。

トピック

- CloudWatch のバケットの Amazon S3 デイリーストレージメトリクス。
- CloudWatch の Amazon S3 リクエストメトリクス
- CloudWatch の S3 レプリケーションメトリクス
- CloudWatch での S3 Storage Lens のメトリクス
- CloudWatch の S3 Object Lambda リクエスト
- CloudWatch のアウトポトスメトリックの Amazon S3
- CloudWatch の Amazon S3 ディメンション
- CloudWatch の S3 レプリケーションディメンション
- CloudWatch の S3 Storage Lens のディメンション。
- CloudWatch の S3 Object Lambda リクエストのディメンション

CloudWatch のバケットの Amazon S3 デイリーストレージメトリクス。

AWS/S3 名前空間には、バケットの以下の日次ストレージメトリクスが含まれます。

メトリクス	説明
BucketSizeBytes	次のストレージクラスのバケットに保存されているデータ量 (バイト単位): • S3 Standard (STANDARD)

メトリクス	説明
	• S3 Intelligent-Tiering (INTELLIGENT_TIERING) • S3 Standard-Infrequent Access (STANDARD_IA) • S3 One Zone-Infrequent Access (ONEZONE_IA) • S3 低冗長ストレージ (RRS) (REDUCED_REDUNDANCY) • S3 Glacier Instant Retrieval (GLACIER_IR) • S3 Glacier Deep Archive (DEEP_ARCHIVE) • S3 Glacier Flexible Retrieval (GLACIER) • S3 Express One Zone (EXPRESS_ONEZONE) この値は、バケット内のすべてのオブジェクト (現行オブジェクトと旧オブジェクトの両方) とメタデータ (バケット名など) のサイズを合計することによって計算され、これにはバケットに対する未完了のすべてのマルチパートアップロードのすべてのパートのサイズも含まれます。 有効なストレージタイプのフィルター: StandardStorage 、IntelligentTieringFAStorage 、IntelligentTieringIAStorage 、IntelligentTieringAAStorage 、IntelligentTieringAIAStorage 、IntelligentTieringDAAStorage 、StandardIAStorage 、StandardIASizeOverhead 、StandardIAObjectOverhead 、OneZoneIAStorage 、OneZoneIASizeOverhead 、ReducedRedundancyStorage 、GlacierInstantRetrievalSizeOverhead GlacierInstantRetrievalStorage 、GlacierStorage 、GlacierStagingStorage 、GlacierObjectOverhead 、GlacierS3ObjectOverhead 、DeepArchiveStorage 、DeepArchiveObjectOverhead 、DeepArchiveS3ObjectOverhead 、DeepArchiveStagingStorage 、ExpressOneZone (「StorageType ディメンション」を参照) 単位: バイト 有効な統計: Average

メトリクス	説明
NumberOfObjects	すべてのストレージクラスのバケットに保存されたオブジェクトの合計数。この値を計算するには、バケット内のすべてのオブジェクト (最新のオブジェクトと最新でないオブジェクトの両方)、削除マーカー、およびバケットに対するすべての不完全なマルチパートアップロードの合計パート数をカウントします。S3 Express One Zone ストレージ クラスのオブジェクトがあるディレクトリバケットの場合、この値はバケット内のすべてのオブジェクトをカウントすることによって算出されます。ただし、バケットへの完了しなかった複数のアップロードは含まれません。 有効なストレージタイプのフィルタ: AllStorageTypes (StorageType ディメンションを参照) 単位: カウント 有効な統計: Average

CloudWatch の Amazon S3 リクエストメトリクス

AWS/S3 名前空間には、次のリクエストメトリクスが含まれます。これらのメトリクスには、請求対象外のリクエスト (CopyObject および Replication からの GET リクエストの場合) が含まれます。

> **Note**
>
> CloudWatch の Amazon S3 リクエストメトリクスは、ディレクトリバケットではサポートされません。

メトリクス	説明
AllRequests	タイプに関係なく、Amazon S3 バケットに対して行われた HTTP リクエストの総数。フィルターでメトリクス設定を使用している場合、このメトリクスはフィルターの要件を満たす HTTP リクエストのみを返します。 単位: カウント

メトリクス	説明
	有効な統計: Sum
GetRequests	Amazon S3 バケット内のオブジェクトに対して行われた HTTP GET リクエストの数。これには、リストオペレーションは含まれません。このメトリクスは、CopyObject リクエストのリクエスト元ごとに増加します。 単位: カウント 有効な統計: Sum **Note** ListMultipartUploads、ListParts、ListObjectVersions など、ページ分割されたリスト指向のリクエストは、このメトリクスに含まれません。
PutRequests	Amazon S3 バケット内のオブジェクトに対して行われた HTTP PUT リクエストの数。このメトリクスは、CopyObject リクエストのリクエスト先ごとに増加します。 単位: カウント 有効な統計: Sum
DeleteRequests	Amazon S3 バケット内のオブジェクトに対して行われた HTTP DELETE リクエストの数。このメトリクスには DeleteObjects リクエストも含まれます。このメトリクスは、削除されるオブジェクトの数ではなく作成されたリクエストの数を示します。 単位: カウント 有効な統計: Sum

メトリクス	説明
HeadRequests	Amazon S3 バケットに対して行われた HTTP HEAD リクエストの数。 単位: カウント 有効な統計: Sum
PostRequests	Amazon S3 バケットに対して行われた HTTP POST リクエストの数。 単位: カウント 有効な統計: Sum Note DeleteObjects および SelectObjectContent リクエストはこのメトリクスに含まれません。
SelectRequests	Amazon S3 バケット内のオブジェクトに対して行われた Amazon S3 の SelectObjectContent リクエストの数。 単位: カウント 有効な統計: Sum
SelectBytesScanned	Amazon S3 バケットの Amazon S3 の SelectObjectContent リクエストでスキャンされたデータのバイト数。 単位: バイト 有効な統計: Average (リクエストあたりのバイト数)、Sum (期間あたりのバイト数)、Sample Count、Min、Max (p100 と同じ)、p0.0 ~ p99.9 のパーセンタイル

メトリクス	説明
SelectBytesReturned	Amazon S3 バケットの Amazon S3 の SelectObjectContent リクエストで返されたデータのバイト数。 単位: バイト 有効な統計: Average (リクエストあたりのバイト数)、Sum (期間あたりのバイト数)、Sample Count、Min、Max (p100 と同じ)、p0.0 ~ p99.9 のパーセンタイル
ListRequests	バケットの内容をリストする HTTP リクエストの数。 単位: カウント 有効な統計: Sum
BytesDownloaded	Amazon S3 バケットに対する、レスポンスに本文が含まれるリクエストに対してダウンロードしたバイト数。 単位: バイト 有効な統計: Average (リクエストあたりのバイト数)、Sum (期間あたりのバイト数)、Sample Count、Min、Max (p100 と同じ)、p0.0 ~ p99.9 のパーセンタイル
BytesUploaded	Amazon S3 バケットに対する、リクエストに本文が含まれるリクエストに対してアップロードしたバイト数。 単位: バイト 有効な統計: Average (リクエストあたりのバイト数)、Sum (期間あたりのバイト数)、Sample Count、Min、Max (p100 と同じ)、p0.0 ~ p99.9 のパーセンタイル

メトリクス	説明
4xxErrors	Amazon S3 バケットに対して行われた、値が 0 または 1 の HTTP 4xx クライアントエラーステータスコードリクエストの数。AAverage 統計はエラーレートを示し、Sum 統計は各期間中のそのタイプのエラー数を示します。 単位: カウント 有効な統計: Average (リクエストあたりの報告数)、Sum (期間あたりの報告数)、Min、Max、Sample Count
5xxErrors	Amazon S3 バケットに対して行われた、値が 0 または 1 の HTTP 5xx サーバーエラーステータスコードリクエストの数。AAverage 統計はエラーレートを示し、Sum 統計は各期間中のそのタイプのエラー数を示します。 単位: カウント 有効な統計: Average (リクエストあたりの報告数)、Sum (期間あたりの報告数)、Min、Max、Sample Count
FirstByteLatency	Amazon S3 バケットがリクエスト全体を受信してからレスポンスの返信が開始するまでのリクエストあたりの時間。 単位: ミリ秒 有効な統計: Average、Sum、Min、Max (p100 と同じ)、Sample Count、p0.0 ~ p100 のパーセンタイル
TotalRequestLatency	最初のバイトが受信されてから Amazon S3 バケットに最後のバイトが送信されるまでのリクエストあたりの経過時間。このメトリクスには、FirstByteLatency には含まれない、リクエストボディの受信とレスポンス本文の送信にかかった時間が含まれます。 単位: ミリ秒 有効な統計: Average、Sum、Min、Max (p100 と同じ)、Sample Count、p0.0 ~ p100 のパーセンタイル

CloudWatch の S3 レプリケーションメトリクス

保留中のバイト数、保留中のオペレーション、およびレプリケーションのレイテンシーを追跡することにより、S3 レプリケーションメトリクスを使用してレプリケーションの進行状況をモニタリングできます。詳細については、「レプリケーションメトリクスによる進捗状況のモニタリング」をご参照ください。

> **Note**
>
> Amazon CloudWatch でレプリケーションメトリクスのアラームを有効にできます。レプリケーションメトリクスのアラームを設定する場合は、[Missing data treatment (欠落データの処理)] フィールドを [Treat missing data as ignore (maintain the alarm state) (欠落データの処理方法: 無視 (アラームの状態を維持)] に設定します。

メトリクス	説明
`ReplicationLatency`	特定のレプリケーションルールについて、レプリケート先 AWS リージョン がレプリケート元 AWS リージョン より遅れる最大秒数。 単位: 秒 有効な統計: Max
`BytesPendingReplication`	特定のレプリケーションルールについて、レプリケーションが保留中のオブジェクトの合計バイト数。 単位: バイト 有効な統計: Max
`OperationsPendingReplication`	特定のレプリケーションルールについて、レプリケーションが保留中のオペレーションの数。 単位: カウント 有効な統計: Max

メトリクス	説明
`OperationsFailedReplication`	特定のレプリケーションルールについて、レプリケートに失敗したオペレーションの数。 単位: カウント 有効な統計: 合計 (失敗した操作の総数)、平均 (失敗率)、サンプル数 (レプリケーションオペレーションの総数)

CloudWatch での S3 Storage Lens のメトリクス

S3 StorageLens の使用状況とアクティビティのメトリクスを Amazon CloudWatch に公開して、CloudWatch ダッシュボードで運用状態の統一されたビューを作成できます。CloudWatch 内の `AWS/S3/Storage-Lens` 名前空間 S3 Storage Lens メトリクスを公開します。CloudWatch 公開オプションは、アドバンストメトリクスとレコメンデーションにアップグレードされている S3 StorageLens ダッシュボードで利用できます。

CloudWatch に公開された S3 Storage Lens メトリクスの一覧については、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。ディメンションの詳細なリストについては、ディメンション を参照してください。

CloudWatch の S3 Object Lambda リクエスト

S3 Object Lambda には、次のリクエストメトリクスが含まれます。

メトリクス	説明
`AllRequests`	Object Lambda アクセスポイントを使用して Amazon S3 バケットに対して行われた HTTP リクエストの総数。 単位: カウント 有効な統計: Sum
`GetRequests`	Object Lambda アクセスポイントを使用してオブジェクトに対して行われた HTTP `GET` リクエストの数。このメトリクスにはリストオペレーションは含まれません。

メトリクス	説明
	単位: カウント 有効な統計: Sum
BytesUploaded	Object Lambda アクセスポイントを使用して Amazon S3 バケットにアップロードされたバイト数。リクエストには本文が含まれます。 単位: バイト 有効な統計: Average (リクエストあたりのバイト数)、Sum (期間あたりのバイト数)、Sample Count、Min、Max (p100 と同じ)、p0.0 ~ p99.9 のパーセンタイル
PostRequests	Object Lambda アクセスポイントを使用して Amazon S3 バケットに対して行われた HTTP POST リクエストの数。 単位: カウント 有効な統計: Sum
PutRequests	Object Lambda アクセスポイントを使用して Amazon S3 バケット内のオブジェクトに対して行われた HTTP PUT リクエストの数。 単位: カウント 有効な統計: Sum
DeleteRequests	Object Lambda アクセスポイントを使用して Amazon S3 バケット内のオブジェクトに対して行われた HTTP DELETE リクエストの数。このメトリクスには DeleteObjects リクエストが含まれます。このメトリクスは、削除されるオブジェクトの数ではなく作成されたリクエストの数を示します。 単位: カウント 有効な統計: Sum

メトリクス	説明
BytesDownloaded	Amazon S3 バケットに対する、レスポンスに本文が含まれる Object Lambda のアクセスポイントを使用したリクエストに対してダウンロードしたバイト数。 単位: バイト 有効な統計: Average (リクエストあたりのバイト数)、Sum (期間あたりのバイト数)、Sample Count、Min、Max (p100 と同じ)、p0.0 ~ p99.9 のパーセンタイル
FirstByteLatency	Amazon S3 バケットが Object Lambda を経由してリクエスト全体を受信してからレスポンスの返信が開始するまでのリクエストあたりの時間。このメトリクスは、AWS Lambda 関数が Object Lambda アクセスポイントにバイトを返すまでにオブジェクトを変換する関数の実行時間によって異なります。 単位: ミリ秒 有効な統計: Average、Sum、Min、Max (p100 と同じ)、Sample Count、p0.0 ~ p100 のパーセンタイル
TotalRequestLatency	最初のバイトが受信されてから Object Lambda アクセスポイントに最後のバイトが送信されるまでのリクエストあたりの経過時間。このメトリクスには、FirstByteLatency には含まれない、リクエストボディの受信とレスポンス本文の送信にかかった時間が含まれます。 単位: ミリ秒 有効な統計: Average、Sum、Min、Max (p100 と同じ)、Sample Count、p0.0 ~ p100 のパーセンタイル
HeadRequests	Object Lambda アクセスポイントを使用して Amazon S3 バケットに対して行われた HTTP HEAD リクエストの数。 単位: カウント 有効な統計: Sum

メトリクス	説明
ListRequests	Amazon S3 バケットの内容をリストする HTTP GET リクエストの数。このメトリクスには、ListObjects および ListObjectsV2 オペレーションの両方が含まれます。 単位: カウント 有効な統計: Sum
4xxErrors	Object Lambda アクセスポイントを使用して Amazon S3 バケットに対して行われた、値が 0 または 1 の HTTP 4xx クライアントエラーステータスコードリクエストの数。AAverage 統計はエラーレートを示し、Sum 統計は各期間中のそのタイプのエラー数を示します。 単位: カウント 有効な統計: Average (リクエストあたりの報告数)、Sum (期間あたりの報告数)、Min、Max、Sample Count
5xxErrors	Object Lambda アクセスポイントを使用して Amazon S3 バケットに対して行われた、値が 0 または 1 の HTTP 5xx サーバーエラーステータスコードリクエストの数。AAverage 統計はエラーレートを示し、Sum 統計は各期間中のそのタイプのエラー数を示します。 単位: カウント 有効な統計: Average (リクエストあたりの報告数)、Sum (期間あたりの報告数)、Min、Max、Sample Count
ProxiedRequests	標準の Amazon S3 API レスポンスを返す Object Lambda アクセスポイントへの HTTP リクエストの数。(このようなリクエストには Lambda 関数は設定されていません。) 単位: カウント 有効な統計: Sum

メトリクス	説明
InvokedLambda	Lambda 関数が呼び出された S3 オブジェクトへの HTTP リクエストの数。 単位: カウント 有効な統計: Sum
LambdaResponseRequests	Lambda 関数によって行われた WriteGetObjectResponse リクエストの数。このメトリクスは、GetObject リクエストにのみ適用されます。
LambdaResponse4xx	Lambda 関数から WriteGetObjectResponse を呼び出すときに発生する HTTP 4xx クライアントエラーの数。このメトリクスは 4xxErrors 呼び出しと同じ情報を提供しますが、対象は WriteGetObjectResponse 呼び出しのみです。
LambdaResponse5xx	Lambda 関数から WriteGetObjectResponse を呼び出すときに発生する HTTP 5xx サーバーエラーの数。このメトリクスは 5xxErrors 呼び出しと同じ情報を提供しますが、対象は WriteGetObjectResponse 呼び出しのみです。

CloudWatch のアウトポストスメトリックの Amazon S3

S3 on Outposts バケットに使用される CloudWatch のメトリクスのリストについては、CloudWatch メトリクス を参照してください。

CloudWatch の Amazon S3 ディメンション

以下のディメンションは、Amazon S3 メトリクスをフィルタリングするために使用されます。

ディメンション	説明
BucketName	このディメンションを指定すると、リクエストしたデータがフィルタリングされて、指定のバケットのものだけになります。

ディメンション	説明
StorageType	このディメンションは、バケットに保存されたデータを、以下のストレージタイプでフィルタリングします。 • StandardStorage - STANDARD ストレージクラスのオブジェクトに使用されているバイト数。 • IntelligentTieringAAStorage - INTELLIGENT_TIERING ストレージクラスのアーカイブアクセス層にあるオブジェクトに使用されているバイト数。 • IntelligentTieringAIAStorage - INTELLIGENT_TIERING ストレージクラスの アーカイブインスタントアクセス層のオブジェクトに使用されているバイト数。 • IntelligentTieringDAAStorage - INTELLIGENT_TIERING ストレージクラスのアーカイブアクセス層にあるオブジェクトに使用されているバイト数。 • IntelligentTieringFAStorage - INTELLIGENT_TIERING ストレージクラスの高頻度アクセス層のオブジェクトに使用されているバイト数。 • IntelligentTieringIAStorage - INTELLIGENT_TIERING ストレージクラスの低頻度アクセス層のオブジェクトに使用されているバイト数。 • StandardIAStorage – S3 標準低頻度アクセス (STANDARD_IA) ストレージクラスのオブジェクトに使用されているバイト数。 • StandardIASizeOverhead – STANDARD_IA ストレージクラスの 128 KB 未満のオブジェクトに使用されているバイト数。 • IntAAObjectOverhead - アーカイブアクセス層の INTELLIGENT_TIERING ストレージクラスの各オブジェクトについて、S3 Glacier はインデックスと関連メタデータ用に 32 KB のストレージを追加します。この追加データは、オブジェクトを特定して復元するのに必要です。この追加ストレージに対しては、S3 Glacier Flexible Retrieval のレートが課金されます。

ディメンション	説明
	• IntAAS3ObjectOverhead - アーカイブアクセス層の INTELLIGENT_TIERING ストレージクラスの各オブジェクトについて、Simple Storage Service (Amazon S3) はオブジェクトの名前とその他のメタデータに 8 KB のストレージを使用します。この追加のストレージに対しては、S3 Standard 料金が発生します。 • IntDAAObjectOverhead - ディープアーカイブアクセス階層の INTELLIGENT_TIERING ストレージクラスの各オブジェクトについて、S3 Glacier はインデックスと関連メタデータ用に 32 KB のストレージを追加します。この追加データは、オブジェクトを特定して復元するのに必要です。この追加ストレージに対しては、S3 Glacier Deep Archive のストレージ料金が課金されます。 • IntDAAS3ObjectOverhead - ディープアーカイブアクセス階層の INTELLIGENT_TIERING ストレージクラスの各オブジェクトについて、Amazon S3 はインデックスと関連メタデータ用に 8 KB のストレージを追加します。この追加データは、オブジェクトを特定して復元するのに必要です。この追加のストレージに対しては、S3 Standard 料金が発生します。 • OneZoneIAStorage - S3 One Zone 低頻度アクセス (ONEZONE_IA) ストレージクラスのオブジェクトに使用されているバイト数。 • OneZoneIASizeOverhead – ONEZONE_IA ストレージクラスの 128 KB 未満のオブジェクトに使用されているバイト数。 • ReducedRedundancyStorage – 低冗長化ストレージ (RRS) クラスのオブジェクトに使用されているバイト数。 • GlacierInstantRetrievalSizeOverhead – S3 Glacier Instant Retrieval ストレージクラスの 128 KB より小さなオブジェクトに使用されているバイト数。

ディメンション	説明
	• `GlacierInstantRetrievalStorage` – S3 Glacier Instant Retrieval ストレージクラスのオブジェクトに使用されているバイト数。 • `GlacierStorage` – S3 Glacier Flexible Retrieval ストレージクラスのオブジェクトに使用されているバイト数。 • `GlacierStagingStorage` - S3 Glacier Flexible Retrieval ストレージクラスのオブジェクトの `CompleteMultipartUpload` リクエストが完了する前に、マルチパートオブジェクトのパートに使用されていたバイト数。 • `GlacierObjectOverhead` - アーカイブされたオブジェクトごとに、S3 Glacier は、インデックスおよび関連するメタデータ用に 32 KB のストレージを追加します。この追加データは、オブジェクトを特定して復元するのに必要です。この追加ストレージに対しては、S3 Glacier Flexible Retrieval のレートが課金されます。 • `GlacierS3ObjectOverhead` - S3 Glacier Flexible Retrieval にアーカイブされたオブジェクトごとに、Simple Storage Service (Amazon S3) はオブジェクトの名前とその他のメタデータに 8 KB のストレージを使用します。この追加のストレージに対しては、S3 Standard 料金が発生します。 • `DeepArchiveStorage` – S3 Glacier ディープアーカイブストレージクラスのオブジェクトに使用されているバイト数。 • `DeepArchiveObjectOverhead` - アーカイブされたオブジェクトごとに、S3 Glacier は、インデックスおよび関連するメタデータ用に 32 KB のストレージを追加します。この追加データは、オブジェクトを特定して復元するのに必要です。この追加ストレージに対しては、S3 Glacier または S3 Glacier Deep Archive の料金が課金されます。 • `DeepArchiveS3ObjectOverhead` – S3 Glacier ディープアーカイブにアーカイブされたオブジェクトごとに、Simple Storage Service (Amazon S3) はオブジェクトの名前とその他のメタデータに 8 KB のストレージを使用しま

ディメンション	説明
	す。この追加のストレージに対しては、S3 Standard 料金が発生します。 • DeepArchiveStagingStorage – S3 Glacier Deep Archive ストレージクラスのオブジェクトの CompleteMultipartUpload (複数アップロード完了) リクエストが完了する前に、マルチパートオブジェクトに使用されているバイト数。 • ExpressOneZone – S3 Express One Zone ストレージクラスのオブジェクトに使用されているバイト数。
FilterId	このディメンションは、バケットに対するリクエストメトリクスに対して指定するメトリクス設定をフィルタリングします。メトリクス設定を作成するときは、フィルター ID (プレフィックス、タグ、アクセスポイントなど) を指定します。詳細については、メトリクス設定の作成を参照してください。

CloudWatch の S3 レプリケーションディメンション

以下のディメンションは、S3 レプリケーションメトリクスをフィルタリングするために使用されます。

ディメンション	説明
SourceBucket	オブジェクトのレプリケート元となるバケットの名前。
DestinationBucket	オブジェクトのレプリケート先となるバケットの名前。
RuleId	このレプリケーションメトリクスの更新をトリガーしたルールの一意の識別子。

CloudWatch の S3 Storage Lens のディメンション。

CloudWatch で S3 Storage Lens メトリクスのフィルタリングに使用されるディメンションのリストについては、「ディメンション」を参照してください。

CloudWatch の S3 Object Lambda リクエストのディメンション

次のディメンションを使用して、Object Lambda アクセスポイントからのデータをフィルタリングします。

ディメンション	説明
`AccessPointName`	リクエストが行われているアクセスポイントの名前。
`DataSourceARN`	Object Lambda アクセスポイントがデータを取得しているソース。リクエストが Lambda 関数を呼び出す場合、これは Lambda Amazon リソースネーム (ARN) を指します。それ以外の場合は、アクセスポイント ARN を指します。

CloudWatch メトリクスへのアクセス

以下の手順を使用して、Amazon S3 のストレージメトリクスを表示できます。Amazon S3 メトリクスを含めるには、開始と終了のタイムスタンプを設定する必要があります。特定の 24 時間あたりのメトリクスの場合は、期間を 1 日の秒数である 86400 秒に設定します。また、`BucketName` および `StorageType` ディメンションを忘れずに設定してください。

AWS CLI の使用

例えば、AWS CLI を使用して特定のバケットの平均サイズ (バイト単位) を取得する場合、次のコマンドを使用できます。

```
aws cloudwatch get-metric-statistics --metric-name BucketSizeBytes --namespace AWS/S3
 --start-time 2016-10-19T00:00:00Z --end-time 2016-10-20T00:00:00Z --statistics Average
 --unit Bytes --region us-west-2 --dimensions Name=BucketName,Value=DOC-EXAMPLE-BUCKET
 Name=StorageType,Value=StandardStorage --period 86400 --output json
```

この例では、次の出力が生成されます。

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-10-19T00:00:00Z",
            "Average": 1025328.0,
            "Unit": "Bytes"
```

```
        }
    ],
    "Label": "BucketSizeBytes"
}
```

S3 コンソールの使用

Amazon CloudWatch コンソールを使用してメトリクスを表示するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 左のナビゲーションペインで [Metrics] (メトリクス) を選択します。
3. [S3] 名前空間を選択します。
4. (オプション) メトリクスを表示するには、検索ボックスにメトリクス名を入力します。
5. (オプション) [StorageType] ディメンションでフィルタする場合は、検索ボックスにストレージクラスの名前を入力します。

AWS CLI を使用して AWS アカウント に保存された有効なメトリクスのリストを表示するには

- コマンドプロンプトで、以下のコマンドを使用します。

```
aws cloudwatch list-metrics --namespace "AWS/S3"
```

CloudWatch のアクセス許可の詳細については「Amazon CloudWatch ユーザーガイド」の「CloudWatch ダッシュボード許可の更新」を参照してください。

CloudWatch メトリクスの設定

Amazon S3 の Amazon CloudWatch リクエストメトリクスを使うと、1 分の CloudWatch メトリクスの受信、CloudWatch アラームの設定、および CloudWatch ダッシュボードへのアクセスによって、Amazon S3 ストレージの運用とパフォーマンスをほぼリアルタイムで確認できます。クラウドストレージに依存するアプリケーションの場合、これらのメトリクスを使用すると、オペレーションの問題を迅速に特定して対応できます。有効になっている場合、デフォルトでは、Amazon S3 バケットレベルでこれらの 1 分のメトリクスが利用可能です。

バケット内のオブジェクトに対する CloudWatch リクエストメトリクスを取得するには、そのバケットのメトリクス設定を作成する必要があります。詳細については、「バケット内のすべてのオブジェクトに対する CloudWatch メトリクス設定を作成する」を参照してください。

共有プレフィックス、オブジェクトタグ、またはアクセスポイントを使用して、収集したメトリクスのフィルタを定義することもできます。フィルタを定義するこの方法を使用すると、メトリクスフィルタを特定のビジネスアプリケーション、ワークフロー、または内部組織に合わせることができます。詳細については、「プレフィックス、オブジェクトタグ、またはアクセスポイントでのメトリクス設定の作成」を参照してください。使用可能な CloudWatch メトリクスとストレージおよびリクエストメトリクスの違いの詳細については、Amazon CloudWatch によるメトリクスのモニタリング を参照してください。

メトリクス設定を使用するときは、以下に注意してください。

- バケットあたりの最大メトリクス設定数は 1000 です。
- フィルタを使用して、メトリクス設定に含めるバケット内のオブジェクトを選択できます。共有プレフィックス、オブジェクトタグ、またはアクセスポイントでフィルタリングすると、メトリクスを特定のビジネスアプリケーション、ワークフロー、または内部組織に合わせることができます。バケット全体のメトリクスをリクエストするには、フィルタのないメトリクス設定を作成します。
- メトリクス設定は、リクエストメトリクスを有効にするためにのみ必要です。バケットレベルでの日次ストレージメトリクスは常にオンになり、追加料金なしで提供されます。現在、フィルタリングされたオブジェクトサブセットの日次ストレージメトリクスを取得することはできません。
- 各メトリクス設定では、使用可能なリクエストメトリクスのフルセットが有効になります。オペレーション固有のメトリクス (`PostRequests` など) は、バケットまたはフィルタにそのタイプのリクエストがある場合にのみ報告されます。
- リクエストメトリクスは、オブジェクトレベルのオペレーションに対して報告されます。それらは、GET Bucket (List Objects)、GET Bucket Object Versions、および マルチパートアップロードのリストなど、バケット内容を一覧表示するオペレーションについても報告されますが、その他のバケットオペレーションでは報告されません。
- リクエストメトリクスでは、プレフィックス、オブジェクトタグ、またはアクセスポイントによるフィルタリングがサポートされますが、ストレージメトリクスではサポートされません。

ベストエフォート型の CloudWatch メトリクス配信

CloudWatch メトリクスは、ベストエフォートで配信されます。リクエストメトリクスを持つ Amazon S3 オブジェクトのほとんどのリクエストにより、データポイントが CloudWatch に送信されます。

メトリクスの完全性や適時性は保証されません。特定のリクエストのデータポイントが、リクエストが実際に処理されたよりも後のタイムスタンプで返される場合があります。1 分のデータポイントは CloudWatch を通じて利用可能になるまで遅延する場合や、まったく提供されない場合がありま

す。CloudWatch リクエストメトリクスは、バケットに対するトラフィックの特性をほぼリアルタイムで示します。すべてのリクエストを完全に報告するためのものではありません。

この機能はベストエフォート型であるため、請求情報とコスト管理ダッシュボードで利用できるレポートには、バケットメトリクスに表示されない 1 つ以上のアクセスリクエストが含まれることがあります。

Amazon S3 での CloudWatch メトリクスの操作方法の詳細については、以下のトピックを参照してください。

トピック

- バケット内のすべてのオブジェクトに対する CloudWatch メトリクス設定を作成する
- プレフィックス、オブジェクトタグ、またはアクセスポイントでのメトリクス設定の作成
- メトリクスフィルタの削除

バケット内のすべてのオブジェクトに対する CloudWatch メトリクス設定を作成する

リクエストメトリクスを設定するときは、バケット内のすべてのオブジェクトに対して CloudWatch メトリクス設定を作成するか、プレフィックス、オブジェクトタグ、またはアクセスポイントでフィルタリングすることができます。このトピックの手順は、バケット内のすべてのオブジェクトに対して設定を作成する方法を説明します。オブジェクトタグ、プレフィックス、またはアクセスポイントでフィルタリングする設定を作成するには、プレフィックス、オブジェクトタグ、またはアクセスポイントでのメトリクス設定の作成 を参照してください。

Amazon S3 の Amazon CloudWatch メトリクスには、ストレージメトリクス、リクエストメトリクス、レプリケーションメトリクスの 3 種類があります。ストレージメトリクスは、1 日に 1 回レポートされ、すべてのお客様に追加料金なしで提供されます。リクエストメトリクスは、処理のレイテンシーの後に 1 分間隔で使用できます。リクエストメトリクスには、CloudWatch の標準料金が課金されます。コンソールで設定するか、Amazon S3 API を使用して、リクエストメトリクスを取得する必要があります。S3 レプリケーションメトリクス は、レプリケーション設定のレプリケーションルールの詳細なメトリクスを提供します。レプリケーションメトリクスを使って、保留中のバイト数、保留中のオペレーション、レプリケーションに失敗したオペレーション、レプリケーションのレイテンシーを追跡すると、レプリケーションの進行状況を 1 分単位でモニタリングできます。

Amazon S3 の CloudWatch メトリクスの詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。

Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、または Amazon S3 REST API を使用して、バケットにメトリクス設定を追加できます。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets] (バケット) リストで、リクエストメトリクスを取得するオブジェクトが含まれているバケットの名前を選択します。
3. [Metrics] (メトリクス) タブをクリックします。
4. [バケットメトリクス] で、[その他のグラフを表示] をクリックします。
5. [リクエストメトリクス] タブをクリックします。
6. [フィルターの作成] をクリックします。
7. [フィルタ名] ボックスに、フィルタ名を入力します。

   名前には、文字、数字、ピリオド、ダッシュ、アンダースコアのみを使用できます。すべてのオブジェクトに適用されるフィルタには、名前 `EntireBucket` を使用することをお勧めします。
8. [フィルタ範囲] で、[このフィルタはバケット内のすべてのオブジェクトに適用されます] を選択します。

   フィルタを定義することで、バケット内のオブジェクトのサブセットに関するメトリクスのみを収集してレポートすることもできます。詳細については、「プレフィックス、オブジェクトタグ、またはアクセスポイントでのメトリクス設定の作成」を参照してください。
9. [Save changes] (変更の保存) をクリックします。
10. [リクエストメトリクス] タブの [フィルタ] で、作成したフィルタを選択します。

    約 15 分後、CloudWatch はこれらのリクエストメトリクスの追跡を開始します。これらは、[リクエストメトリクス] タブで確認できます。メトリクスのグラフは、Amazon S3 または CloudWatch コンソールでレプリケーションメトリクスで表示できます。リクエストメトリクスには、CloudWatch の標準料金が課金されます。詳細については、「Amazon CloudWatch の料金」を参照してください。

REST API の使用

Amazon S3 REST API でメトリクス設定をプログラムで追加することもできます。メトリクス設定の追加と使用の詳細については、「Amazon Simple Storage Service API リファレンス」で以下のトピックを参照してください。

- PUT Bucket Metric Configuration

- GET Bucket Metric Configuration
- List Bucket Metric Configuration
- DELETE Bucket Metric Configuration

AWS CLI の使用

1. AWS CLI をインストールしてセットアップします。手順については、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」を参照してください。
2. ターミナルを開きます。
3. 以下のコマンドを実行してメトリクス設定を追加します。

```
aws s3api put-bucket-metrics-configuration --endpoint https://s3.us-
west-2.amazonaws.com --bucket bucket-name --id metrics-config-id --metrics-
configuration '{"Id":"metrics-config-id"}'
```

プレフィックス、オブジェクトタグ、またはアクセスポイントでのメトリクス設定の作成

Amazon S3 の Amazon CloudWatch メトリクスには、ストレージメトリクス、リクエストメトリクス、レプリケーションメトリクスの 3 種類があります。ストレージメトリクスは、1 日に 1 回レポートされ、すべてのお客様に追加料金なしで提供されます。リクエストメトリクスは、処理のレイテンシーの後に 1 分間隔で使用できます。リクエストメトリクスには、CloudWatch の標準料金が課金されます。コンソールで設定するか、Amazon S3 API を使用して、リクエストメトリクスを取得する必要があります。S3 レプリケーションメトリクス は、レプリケーション設定のレプリケーションルールの詳細なメトリクスを提供します。レプリケーションメトリクスを使って、保留中のバイト数、保留中のオペレーション、レプリケーションに失敗したオペレーション、レプリケーションのレイテンシーを追跡すると、レプリケーションの進行状況を 1 分単位でモニタリングできます。

Amazon S3 の CloudWatch メトリクスの詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。

CloudWatch メトリクスを設定する場合、バケット内のすべてのオブジェクトに対してフィルタを作成することも、1 つのバケット内の関連オブジェクトのグループに設定をフィルタリングすることもできます。以下のフィルタタイプの 1 つ以上に基づいて、メトリクス設定に含めるバケット内のオブジェクトをフィルタリングできます。

- オブジェクトキー名のプレフィックス – Amazon S3 データモデルはフラット構造ですが、プレフィックスを使用して階層を推測できます。Amazon S3 コンソールはフォルダの概念でこれらのプレフィックスをサポートします。プレフィックスでフィルタリングした場合、同じプレフィックスを持つオブジェクトがメトリクス設定に含まれます。 プレフィックスの詳細については、プレフィックスを使用してオブジェクトを整理する を参照してください。
- タグ – オブジェクトにタグ (キー値と名前のペア) を追加できます。タグを使用すると、オブジェクトを簡単に検索および整理できます。メトリクス設定のフィルタとしてタグを使用することもできます。オブジェクトタグの詳細については、タグを使用してストレージを分類するを参照してください。
- アクセスポイント – S3 アクセスポイントは、バケットに接続され、S3 の共有データセットの大規模なデータアクセスの管理を簡素化する名前付きネットワークエンドポイントです。アクセスポイントフィルタを作成すると、Amazon S3 には、メトリクス設定で指定したアクセスポイントへのリクエストが含まれます。詳細については、「アクセスポイントのモニタリングとログ記録」を参照してください。

> **Note**
>
> アクセスポイントでフィルタリングするメトリクス設定を作成する場合は、アクセスポイントのエイリアスではなく、アクセスポイントの Amazon リソースネーム (ARN) を使用する必要があります。ARN は、特定のオブジェクトの ARN ではなく、アクセスポイント自体に使用してください。アクセスポイントの ARN の詳細については、「アクセスポイントの使用」を参照してください。

フィルタを指定した場合、単一のオブジェクトで機能するリクエストのみがフィルタと一致し、報告されたメトリクスに含められます。DeleteObjects や `ListObjects` のリクエストは、フィルターによる設定のメトリクスを一切返しません。

より複雑なフィルタリングをリクエストするには、複数の要素を選択します。これらのすべての要素を持つオブジェクトのみがメトリクス設定に含められます。フィルタを設定しない場合、バケット内のすべてのオブジェクトがメトリクス設定に含められます。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

2. [Buckets] (バケット) リストで、リクエストメトリクスを取得するオブジェクトが含まれているバケットの名前を選択します。
3. [Metrics] (メトリクス) タブをクリックします。
4. [バケットメトリクス] で、[その他のグラフを表示] をクリックします。
5. [リクエストメトリクス] タブをクリックします。
6. [フィルターの作成] をクリックします。
7. [フィルタ名] ボックスに、フィルタ名を入力します。

   名前には、文字、数字、ピリオド、ダッシュ、アンダースコアのみを使用できます。
8. [フィルタの範囲] で、プレフィックス、オブジェクトタグ、S3 アクセスポイント、または 3 つすべての組み合わせを使用して、このフィルタの範囲を制限します。。
9. [フィルタのタイプで、フィルタタイプをプレフィックス、オブジェクトタグ、またはアクセスポイントから少なくとも 1 つ選択します。
10. [プレフィックス] ボックスにプレフィックスを入力して、プレフィックスフィルタを定義し、フィルタの範囲を単一のパスに制限します。
11. オブジェクトタグフィルタを定義するには、[オブジェクトタグ] で、[タグの追加] を選択し、タグのキーおよび値を入力します。
12. アクセスポイントフィルタを定義するには、[S3 アクセスポイント] フィールドに、アクセスポイント ARN を入力するか、[S3 を参照] を選択して、アクセスポイントに移動します。

    > **Important**
    >
    > アクセスポイントのエイリアスは入力できません。特定のオブジェクトの ARN ではなく、アクセスポイント自体の ARN を入力する必要があります。

13. [Save changes] (変更の保存) をクリックします。

    Amazon S3 は、指定したプレフィックス、タグ、またはアクセスポイントを使用するフィルタを作成します。
14. [リクエストメトリクス] タブの [フィルタ] で、作成したフィルタを選択します。

    これで、リクエストメトリクスの範囲をプレフィックス、タグ、またはアクセスポイントで制限するフィルタが作成されました。CloudWatch がこれらのリクエストメトリクスの追跡を開始してから約 15 分後に、Amazon S3 コンソールと CloudWatch コンソールの両方でメトリクスのグラフを表示できるようになります。リクエストメトリクスには、CloudWatch の標準料金が課金されます。詳細については、「Amazon CloudWatch の料金」を参照してください。

バケットレベルでリクエストメトリクスを設定することもできます。詳細については、バケット内のすべてのオブジェクトに対する CloudWatch メトリクス設定を作成する を参照してください。

AWS CLI の使用

1. AWS CLI をインストールしてセットアップします。手順については、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」を参照してください。
2. ターミナルを開きます。
3. 以下のコマンドを実行してメトリクス設定を追加します。

   Example : プレフィックスでフィルタリングするには

   ```
   aws s3api put-bucket-metrics-configuration --bucket DOC-EXAMPLE-BUCKET1 --
   id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
   {"Prefix":"prefix1"}} '
   ```

   Example : タグでフィルタリングするには

   ```
   aws s3api put-bucket-metrics-configuration --bucket DOC-EXAMPLE-BUCKET1 --
   id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
   {"Tag": {"Key": "string", "Value": "string"}} '
   ```

   Example : アクセスポイントでフィルタリングするには

   ```
   aws s3api put-bucket-metrics-configuration --bucket DOC-EXAMPLE-BUCKET1 --
   id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
   {"AccessPointArn":"arn:aws:s3:Region:account-id:accesspoint/access-point-name"}} '
   ```

   Example : プレフィックス、タグ、アクセスポイントでフィルタリングするには

   ```
   aws s3api put-bucket-metrics-configuration --endpoint https://
   s3.Region.amazonaws.com --bucket DOC-EXAMPLE-BUCKET1 --id metrics-config-id --
   metrics-configuration '
   {
       "Id": "metrics-config-id",
       "Filter": {
   ```

```
        "And": {
            "Prefix": "string",
            "Tags": [
                {
                    "Key": "string",
                    "Value": "string"
                }
            ],
            "AccessPointArn": "arn:aws:s3:Region:account-id:accesspoint/access-
point-name"
        }
    }
}'
```

REST API の使用

Amazon S3 REST API でメトリクス設定をプログラムで追加することもできます。メトリクス設定の追加と使用の詳細については、「Amazon Simple Storage Service API リファレンス」で以下のトピックを参照してください。

- PUT Bucket Metric Configuration
- GET Bucket Metric Configuration
- List Bucket Metric Configuration
- DELETE Bucket Metric Configuration

メトリクスフィルタの削除

Amazon CloudWatch リクエストメトリクスフィルタが不要になった場合は、削除できます。フィルタを削除すると、その特定のフィルタを使用するリクエストメトリクスに対して課金されなくなります。ただし、存在する他のフィルタ設定については、引き続き課金されます。

フィルタを削除すると、リクエストメトリクスにフィルタを使用できなくなります。フィルタを削除すると、元に戻すことはできません。

リクエストメトリクスフィルタの作成については、次のトピックを参照してください。

- バケット内のすべてのオブジェクトに対する CloudWatch メトリクス設定を作成する
- プレフィックス、オブジェクトタグ、またはアクセスポイントでのメトリクス設定の作成

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、バケット名を選択します。
3. [Metrics] (メトリクス) タブをクリックします。
4. [バケットメトリクス] で、[その他のグラフを表示] をクリックします。
5. [リクエストメトリクス] タブをクリックします。
6. [フィルタの管理] を選択します。
7. フィルタを選択します。

   > **Important**
   > フィルタを削除すると、元に戻すことはできません。

8. [削除] を選択します。

   Amazon S3 によってフィルタが削除されます。

REST API の使用

Amazon S3 REST API でメトリクス設定をプログラムで追加することもできます。メトリクス設定の追加と使用の詳細については、「Amazon Simple Storage Service API リファレンス」で以下のトピックを参照してください。

- PUT Bucket Metric Configuration
- GET Bucket Metric Configuration
- List Bucket Metric Configuration
- DELETE Bucket Metric Configuration

Amazon S3 イベント通知

Amazon S3 イベント通知機能を使用して、S3 バケットで特定のイベントが発生したときに通知を受け取ることができます。通知を有効にするには、Amazon S3 から発行するイベントを識別する通知設定を追加します。Amazon S3 から通知を送信する宛先も指定されていることを確認してくださ

い。この設定は、バケットに関連付けられた通知サブリソースに保存します 詳細については、「バケット設定オプション」を参照してください。Amazon S3 では、このサブリソースを管理するための API も利用できます。

> **Important**
>
> Amazon S3 イベント通知は、少なくとも 1 回配信されるように設計されています。通常、イベント通知は数秒で配信されますが、1 分以上かかる場合もあります。

Amazon S3 イベント通知の概要。

現在、Amazon S3 は次のイベントの通知を発行できます。

- 新しいオブジェクトがイベントを作成しました。
- オブジェクトの削除イベント
- オブジェクトイベントの復元
- 低冗長化ストレージ (RRS) オブジェクトがイベントを紛失した
- レプリケーションイベント
- S3 ライフサイクルの有効期限イベント
- S3 ライフサイクルの移行イベント
- S3 Intelligent-Tiering 自動アーカイブイベント
- オブジェクトのタグ付けイベント
- オブジェクト ACL PUT イベント

サポートしたすべてのイベントタイプの詳細は、SQS、SNS、および Lambda でサポートされているイベントタイプ を参照してください。

Amazon S3 は、次の宛先にイベントの通知メッセージを送信できます。通知設定でこれらの送信先の Amazon リソースネーム (ARN) 値を指定します。

- Amazon Simple Notification Service (Amazon SNS)のトピック
- Amazon Simple Queue Service Amazon SQS キュー
- AWS Lambda 関数
- Amazon EventBridge

詳細については、「サポートされているイベントの送信先」を参照してください。

> **Note**
>
> Amazon Simple キューサービス FIFO (先入れ先出し) キューは、Amazon S3 イベント通知の送信先としてサポートされていません。Amazon S3 イベントの通知を Amazon SQS FIFO キューに送信するには、Amazon EventBridge を使用できます。詳細については、「Amazon EventBridge を有効にします」を参照してください。

> **Warning**
>
> 通知が通知をトリガーするのと同じバケットに書き込むと、実行ループが発生する可能性があります。例えば、オブジェクトがアップロードされるたびにバケットで Lambda 関数をトリガーし、その関数によってオブジェクトがバケットにアップロードされると、その関数によって間接的にその関数自体がトリガーされます。これを回避するには、2 つのバケットを使用するか、受信オブジェクトで使用されるプレフィックスにのみ適用されるようにトリガーを設定します。
>
> AWS Lambda で Amazon S3 通知を使用する方法の詳細と例については、「AWS Lambda デベロッパーガイド」の「AWS Lambda を Amazon S3 に使用する」を参照してください。

バケットごとに作成できるイベント通知設定数の詳細については、AWS 全般のリファレンスの「Amazon S3 サービスクォータ」を参照してください。

イベント通知の詳細については、以下のセクションを参照してください。

トピック

- イベント通知のタイプおよび送信先
- Amazon SQS、Amazon SNS、Lambda を使用します
- EventBridge の使用

イベント通知のタイプおよび送信先

Amazon S3 では、通知を発行できるいくつかのイベント通知のタイプと送信先がサポートされています。イベント通知を設定するときに、イベントタイプと送信先を指定できます。各イベント通知に

指定できる送信先は 1 つだけです。Amazon S3 イベント通知は、通知メッセージごとに 1 つのイベントエントリを送信します。

トピック

サポートされているイベントの送信先

Amazon S3 は、次の宛先にイベントの通知メッセージを送信できます。

- Amazon Simple Notification Service (Amazon SNS)のトピック
- Amazon Simple Queue Service Amazon SQS キュー
- AWS Lambda
- Amazon EventBridge

ただし、各イベント通知に指定できる送信先タイプは 1 つだけです。

> **Note**
>
> Amazon SNS トピックまたは Amazon SQS キューにメッセージを投稿するには、Amazon S3 のアクセス許可を付与する必要があります。また、ユーザーに代わって AWS Lambda 関数を呼び出すためにも Amazon S3 の許可を付与する必要があります。これらの許可を付与する方法については、宛先にイベント通知メッセージを発行するアクセス許可の付与 を参照してください。

Amazon SNS トピック

Amazon SNS は、柔軟性に優れたフルマネージド型のプッシュメッセージングサービスです。このサービスを使用すると、モバイルデバイスまたは配信サービスにメッセージを配信できます。SNS で 一度メッセージを送信すると、1 回または複数回配信できます。現在 、 標準 SNS は S3 イベント通知の宛先としてのみ許可されていますが、SNS FIFO は許可されていません。

Amazon SNS は承認エンドポイントやクライアントへのメッセージの配信または送信の調整と管理を行います。Amazon SNS コンソールを使用して、通知の送信先にする Amazon SNS トピックを作成できます。

トピックは、Amazon S3 バケットと同じ AWS リージョンにある必要があります。Amazon SNS トピックの作成方法の詳細については、Amazon Simple Notification Service デベロッパーガイドの Amazon SNS の開始方法および Amazon SNS のよくある質問 を参照してください。

イベント通知宛先として作成した Amazon SNS トピックを使用するには、以下のものが必要です。

- Amazon SNS トピックの Amazon リソースネーム (ARN)
- 有効な Amazon SNS トピックサブスクリプション。これにより、Amazon SNS トピックにメッセージが公開されると、トピックのサブスクライバーに通知されます。

Amazon SQS キュー

Amazon SQS には、コンピュータ間で送受信されるメッセージを格納するための、信頼性の高いスケーラブルなホストされたキューが用意されています。Amazon SQS を使用すると、どのような量のデータでも転送することができ、他のサービスが常に利用可能である必要もありません。Amazon SQS コンソールを使用すると、通知の送信先にする Amazon SQS キューを作成できます。

Amazon SQS キューは、Amazon S3 バケットと同じ AWS リージョン に存在する必要があります。Amazon SQS キューの作成方法の説明は、「Amazon Simple Queue Service デベロッパーガイド」の「Amazon Simple Queue Service とは」および「Amazon SQS の開始方法」を参照してください。

イベント通知の送信先として Amazon SQS キューを使用するには、以下が必要です。

- Amazon SQS キューの Amazon リソースネーム (ARN)

> **Note**
>
> Amazon Simple キューサービス FIFO (先入れ先出し) キューは、Amazon S3 イベント通知の送信先としてサポートされていません。Amazon S3 イベントの通知を Amazon SQS FIFO キューに送信するには、Amazon EventBridge を使用できます。詳細については、「Amazon EventBridge を有効にします」を参照してください。

Lambda 関数

AWS Lambda を使用して、AWS の他のサービスをカスタムロジックで拡張したり、AWS のスケール、パフォーマンス、セキュリティで動作する独自のバックエンドを作成したりすることができます。Lambda を使用すると、必要な場合にのみ実行される離散イベント駆動型アプリケーションを作成できます。また、これを使用して、これらのアプリケーションを 1 日数回のリクエストから 1 秒あたり数千回のリクエストに自動的にスケーリングすることもできます。

Lambda は、Amazon S3 バケットイベントに応答してカスタムコードを実行できます。カスタムコードを Lambda にアップロードし、Lambda 関数と呼ばれるものを作成します。Amazon S3 は特定のタイプのイベントを検出すると、そのイベントを AWS Lambda に公開し、Lambda で関数を呼び出すことができます。それに応じて、Lambda が関数を実行します。例えば、検出される可能性のあるイベントタイプの 1 つは、オブジェクトで作成されたイベントです。

AWS Lambda コンソールを使用すると、AWS インフラストラクチャを使用してユーザーに代わってコードを実行する Lambda 関数を作成できます。Lambda 関数は S3 バケットと同じリージョンに存在する必要があります。Lambda 関数をイベント通知の送信先として設定するには、Lambda 関数の名前または ARN も必要になります。

> **Warning**
>
> 通知が通知をトリガーするのと同じバケットに書き込むと、実行ループが発生する可能性があります。例えば、オブジェクトがアップロードされるたびにバケットで Lambda 関数をトリガーし、その関数によってオブジェクトがバケットにアップロードされると、その関数によって間接的にその関数自体がトリガーされます。これを回避するには、2 つのバケットを使用するか、受信オブジェクトで使用されるプレフィックスにのみ適用されるようにトリガーを設定します。
>
> AWS Lambda で Amazon S3 通知を使用する方法の詳細と例については、「AWS Lambda デベロッパーガイド」の「AWS Lambda を Amazon S3 に使用する」を参照してください。

Amazon EventBridge

Amazon EventBridge はサーバーレスのイベントバスで、AWS サービスからのイベントを受信します。イベントに一致するルールを設定し、それらを AWS サービスや HTTP エンドポイントなどのターゲットに配信できます。詳細については、Amazon EventBridge ユーザーガイドの Amazon EventBridge とは を参照してください。

他の宛先とは異なり、バケットの EventBridge へのイベントの配信を有効または無効にすることができます。配信を有効にすると、すべてのイベントが EventBridge に送信されます。さらに、EventBridge ルールを使用して、イベントを追加のターゲットに送信することもできます。

SQS、SNS、および Lambda でサポートされているイベントタイプ

Amazon S3 は、次のタイプのイベントを発行できます。通知設定で、これらのイベントタイプを指定します。

イベントタイプ	説明
s3:TestEvent	通知が有効な場合、Amazon S3 はテスト通知を公開します これは、トピックが存在し、バケット所有者が指定されたトピックを公開する許可があることを保証するためです。 通知の有効化に失敗した場合、テスト通知を受信しません。
s3:ObjectCreated:* s3:ObjectCreated:Put s3:ObjectCreated:Post s3:ObjectCreated:Copy s3:ObjectCreated:CompleteMultipartUpload	PUT、POST、COPY などの Amazon S3 API はオブジェクトを作成できます。これらのイベントタイプで、特定の API を使用してオブジェクトが作成されたときに通知を有効にできます。または、オブジェクトの作成に使用された API に関係なく、`s3:ObjectCreated:*` イベントタイプを使用して通知を要求できます。 `s3:ObjectCreated:CompleteMultipartUpload` には、コピーオペレーションの UploadPartCopy を使用して作成されたオブジェクトが含まれます。
s3:ObjectRemoved:* s3:ObjectRemoved:Delete s3:ObjectRemoved:DeleteMarkerCreated	[ObjectRemoved] イベントタイプを使用することで、オブジェクトまたはオブジェクトのバッチがバケットから削除されたときに通知を有効にできます。 `s3:ObjectRemoved:Delete` イベントタイプを使用することで、オブジェクトが削除されたとき、またはバージョニングが有効なオブジェクトが完全に削除されたときに通知が送信されるようにリクエストできます。その代わりに、`s3:ObjectRemoved:DeleteMarkerCreated` を使用することで、削除マーカーがバージョニング・オ

イベントタイプ	説明
	ブジェクトを作成すると、通知が送信されるようにリクエストできます。バージョン管理されたオブジェクトを削除する方法については、バージョニングが有効なバケットからのオブジェクトバージョンの削除 を参照してください。ワイルドカードとして s3:ObjectRemoved:* を使用することで、オブジェクトが削除されるたびに通知が送信されるようにリクエストできます。 これらのイベント通知では、ライフサイクル設定や失敗したオペレーションから自動削除のアラートはありません。
s3: ObjectRestore: * s3:ObjectRestore:Post s3:ObjectRestore:Completed s3:ObjectRestore:Delete	ObjectRestore イベントタイプを使用して、S3 Glacier Flexible Retrieval ストレージクラスおよび S3 Glacier Deep Archive ストレージクラス、S3 Intelligent-Tiering アーカイブアクセス階層、および S3 Intelligent-Tiering Deep アーカイブアクセス階層からオブジェクトを復元するときに、開始および完了の通知を受け取ることができます。オブジェクトの復元されたコピーの有効期限が切れたときの通知を受け取ることもできます。 s3:ObjectRestore:Post イベントタイプは、オブジェクトの復元開始を通知します。s3:ObjectRestore:Completed イベントタイプは、復元完了を通知します。s3:ObjectRestore:Delete イベントタイプは、復元されたオブジェクトの一時コピーの有効期限が切れると通知します。
s3:ReducedRedundancyLostObject	この通知イベントは、Amazon S3 が RRS ストレージクラスのオブジェクトが失われたことを検出したときに受け取ります。

イベントタイプ	説明
s3:Replication:* s3:Replication:OperationFailedReplication s3:Replication:OperationMissedThreshold s3:Replication:OperationReplicatedAfterThreshold s3:Replication:OperationNotTracked	レプリケーションイベントタイプを使用すると、S3 レプリケーションメトリクスまたは S3 Replication Time Control (S3 RTC) が有効になっているレプリケーション構成の通知を受信できます。保留中のバイト数、保留中のオペレーション、およびレプリケーションのレイテンシーを追跡すると、レプリケーションイベントの進行状況を 1 分単位でモニタリングできます。レプリケーションメトリクスについては、「レプリケーションメトリクスと S3 イベント通知による、進捗状況のモニタリング」をご参照ください。 -s3:Replication:OperationFailedReplication イベントタイプは、レプリケーションのターゲットであったオブジェクトがレプリケートに失敗したときに通知を受け取ります。s3:Replication:OperationMissedThreshold イベントタイプは、レプリケーションに適格なオブジェクトがレプリケーションの 15 分の閾値を超えたときに通知します。 s3:Replication:OperationReplicatedAfterThreshold イベントタイプは、オブジェクトが 15 分の閾値の後に S3 レプリケーションタイムコントロールを使用するレプリケーションが適格であるとき通知します。s3:Replication:OperationNotTracked イベントタイプは、S3 レプリケーションタイムコントロールを使用するレプリケーションに適格であったが、レプリケーションメトリクスによって追跡されなくなったオブジェクトを通知します。

イベントタイプ	説明
s3:LifecycleExpiration:* s3:LifecycleExpiration:Delete s3:LifecycleExpiration:DeleteMarkerCreated	LifecycleExpiration イベントタイプを使用することにより、Amazon S3 が S3 ライフサイクル設定に基づいてオブジェクトを削除したときに通知を受け取ることができます。 `s3:LifecycleExpiration:Delete` イベントタイプは、バージョン管理されていないバケット内のオブジェクトが削除されたときに通知します。また、S3 ライフサイクル設定によってオブジェクトバージョンが完全に削除された場合にも通知されます。`s3:LifecycleExpiration:DeleteMarkerCreated` イベントタイプは、バージョン対応バケット内のオブジェクトの現在のバージョンが削除されたときに S3 ライフサイクル が削除マーカーを作成したときに通知します。
s3:LifecycleTransition	この通知イベントは、S3 ライフサイクル設定によってオブジェクトが別の Amazon S3 ストレージクラスに移行されたときに受信されます。
s3: IntelligentTiering	この通知イベントは、S3 Intelligent-Tiering ストレージクラス内のオブジェクトがアーカイブアクセス層またはディープアーカイブアクセス層に移動したときに受信されます。
s3:ObjectTagging:* s3:ObjectTagging:Put s3:ObjectTagging:Delete	ObjectRemoved イベントタイプを使用することで、オブジェクトタグがオブジェクトに追加またはオブジェクトから削除されたときに通知を有効にできます `s3:ObjectTagging:Put` イベントタイプは、タグがオブジェクトの PUT であるか、既存のタグが更新されたときに通知します。`s3:ObjectTagging:Delete` イベントタイプは、タグがオブジェクトから削除されたときに通知します。

イベントタイプ	説明
s3:ObjectAcl:Put	この通知イベントは、ACL がオブジェクトに PUT されたとき、または既存の ACL が変更されたときに受け取ります。リクエストによってオブジェクトの ACL が変更されない場合、イベントは生成されません。

Amazon EventBridge でサポートされているイベントタイプ。

Amazon S3 から Amazon EventBridge に送信するイベントタイプのリストについては、「EventBridge の使用」を参照してください。

イベントの順序付けと重複イベント

Amazon S3 イベント通知は、通知を少なくとも 1 回配信するように設計されていますが、イベントが発生したのと同じ順序で通知が受信されるとは限りません。まれに、Amazon S3 の再試行メカニズムによって、同じオブジェクトイベントに対して重複する S3 イベント通知が発生することがあります。重複イベントまたは順序が異なるイベント処理の詳細については、「AWS Storage Blog」の「Manage event ordering and duplicate events with Amazon S3 Event Notifications」を参照してください。

Amazon SQS、Amazon SNS、Lambda を使用します

通知の有効化は、バケットレベルのオペレーションです。バケットに関連付けられた 通知 サブリソースに通知設定情報が格納されます。通常、バケット通知設定を作成または変更してから、変更が有効になるまで通常 5 分かかります。通知が最初に有効になったとき、`s3:TestEvent` が発生します。次のいずれかの方法を使用して通知設定の管理を行います。

- Amazon S3 コンソールの使用 – コンソール UI では、コードを記述しなくても、バケットの通知設定を指定できます。詳細については、「Amazon S3 コンソールを使用したイベント通知の有効化と設定」を参照してください。
- AWS SDK をプログラムで使用する – 内部的には、コンソールも SDK も Amazon S3 REST API を呼び出して、バケットに関連付けられた通知サブリソースを管理します。AWS SDK を使用した通知設定の例については、チュートリアル: 通知 (SNS トピックまたは SQS キュー) のバケットを設定する を参照してください。

> **Note**
>
> コードから直接 Amazon S3 REST API を呼び出すこともできます。しかし、そうするのはリクエストを認証するためのコードを作成する必要があるため面倒な場合もあります。

使用する方法を問わず、Amazon S3 は通知設定を XML として、バケットに関連付けられた 通知 サブリソースに保存します。バケットのサブリソースの詳細については、バケット設定オプション を参照してください。

トピック

- 宛先にイベント通知メッセージを発行するアクセス許可の付与
- Amazon S3 コンソールを使用したイベント通知の有効化と設定
- プログラムによるイベント通知の設定
- チュートリアル: 通知 (SNS トピックまたは SQS キュー) のバケットを設定する
- オブジェクトキー名のフィルタリングを使用したイベント通知の設定
- イベントメッセージの構造

宛先にイベント通知メッセージを発行するアクセス許可の付与

Amazon S3 プリンシパルに、SNS トピック、SQS キュー、Lambda 関数へメッセージを公開する関連する API を呼び出すため必要なアクセス許可を付与する必要があります。これは、Amazon S3 がイベント通知メッセージを宛先に公開できるようにするためです。

イベント通知メッセージを送信先に公開する場合のトラブルシューティングについては、「Amazon S3 イベント通知を Amazon Simple Notification Service に公開するためのトラブルシューティング」トピックを参照してください。

トピック

- AWS Lambda 関数を呼び出すアクセス許可の付与
- SNS トピックまたは SQS キューにメッセージを発行するアクセス許可の付与

AWS Lambda 関数を呼び出すアクセス許可の付与

Amazon S3 は Lambda 関数を呼び出してイベントメッセージを AWS Lambda に発行し、イベントメッセージを引数として指定します。

Amazon S3 コンソールを使用して Lambda 関数の AmazonS3 バケットにイベント通知を設定すると、コンソールは Lambda 関数に必要なアクセス許可を設定します。これは、Amazon S3 がバケットから関数を呼び出す許可があるからです。詳細については、「Amazon S3 コンソールを使用したイベント通知の有効化と設定」を参照してください。

AWS Lambda から Amazon S3 の許可を付与して、Lambda 関数を呼び出すこともできます。詳細については、AWS Lambda デベロッパーガイドのチュートリアル: Amazon S3 で AWS Lambda を使用するを参照してください。

SNS トピックまたは SQS キューにメッセージを発行するアクセス許可の付与

SNS トピックや SQS キューにメッセージを発行する許可を Amazon S3 に付与するには、AWS Identity and Access Management (IAM) ポリシーを宛先 SNS トピックや SQS キューにアタッチします。

SNS トピックや SQS キューにポリシーをアタッチする方法の例については、チュートリアル: 通知 (SNS トピックまたは SQS キュー) のバケットを設定する を参照してください。アクセス許可の詳細については、次のトピックを参照してください。

- 「Amazon Simple Notification Service デベロッパーガイド」の「Amazon SNS アクセスコントロールのケース例」
- 「Amazon Simple Queue サービスデベロッパーガイド」の「Amazon SQS での Identity and Access Management」

発行先 SNS トピックの IAM ポリシー

発行先の SNS トピックにアタッチする AWS Identity and Access Management (IAM) ポリシーの例を次に示します。このポリシーを使用してイベント通知の宛先 Amazon SNS トピックを設定する方法については、チュートリアル: 通知 (SNS トピックまたは SQS キュー) のバケットを設定する を参照してください。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
```

```
        {
            "Sid": "Example SNS topic policy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "SNS-topic-ARN",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:bucket-name"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

発行先 SQS キューの IAM ポリシー

以下は、発行先 SQS キューにアタッチする IAM ポリシーの例です。このポリシーを使用してイベント通知の宛先 Amazon SQS キューを設定する方法については、チュートリアル: 通知 (SNS トピックまたは SQS キュー) のバケットを設定する を参照してください。

このポリシーを使用するには、Amazon SQS キューの ARN、バケット名、およびバケット所有者の AWS アカウント ID を更新する必要があります。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SQS:SendMessage"
```

```
            ],
            "Resource": "arn:aws:sqs:Region:account-id:queue-name",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

Amazon SNS および Amazon SQS IAM ポリシーの両方について、StringLike 条件の代わりにポリシーで ArnLike 条件を指定できます。

ArnLike を使用する場合、ARN のパーティション、サービス、アカウント ID、リソースタイプ、および部分的なリソース ID 部分が、リクエストコンテキストの ARN と完全に一致する必要があります。部分一致が許可されるのは、リージョンとリソースパスのみです。

ArnLike の代わりに StringLike を使用すると、マッチングでは ARN 構造が無視され、ワイルドカードで指定された部分に関係なく部分一致が可能になります。詳細については、「IAM ユーザーガイド」の「IAM JSON のポリシー要素」を参照してください。

```
"Condition": {
  "StringLike": { "aws:SourceArn": "arn:aws:s3:*:*:bucket-name" }
  }
```

AWS KMS キーポリシー

SQS キューまたは SNS トピックが AWS Key Management Service (AWS KMS) カスタマーマネージドキーで暗号化されている場合は、暗号化されたトピックまたはキューを操作するための許可を Amazon S3 サービスプリンシパルに付与する必要があります。Amazon S3 サービスプリンシパルに許可を付与するには、カスタマーマネージドキーのキーポリシーに次のステートメントを追加します。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
```

```
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS KMS キーポリシーの詳細については、AWS Key Management Service デベロッパーガイドの AWS KMS でキーポリシーを使用する を参照してください。

Amazon SQS および Amazon SNS 用の AWS KMS でのサーバー側の暗号化の使用の詳細については、以下を参照してください。

- 「Amazon Simple Notification Service デベロッパーガイド」の「キー管理」。
- 「Amazon Simple Queue Service デベロッパーガイド」の「キー管理」。
- AWS KMS コンピューティングブログ の Encrypting messages published to Amazon SNS with AWS。

Amazon S3 コンソールを使用したイベント通知の有効化と設定

イベントが発生するたび、宛先へ通知メッセージを送信する特定の Amazon S3 バケットイベントを有効にできます。このセクションでは、Amazon S3 コンソールを使用してイベント通知を有効にする方法について説明します。AWS SDK および Amazon S3 REST API でイベント通知を使用する方法については、プログラムによるイベント通知の設定 を参照してください。

前提条件: バケットのイベント通知を有効にするには、送信先タイプのいずれか 1 つを設定し、アクセス許可を設定する必要があります。詳細については、サポートされているイベントの送信先および宛先にイベント通知メッセージを発行するアクセス許可の付与を参照してください。

> **Note**
>
> Amazon Simple キューサービス FIFO (先入れ先出し) キューは、Amazon S3 イベント通知の送信先としてサポートされていません。Amazon S3 イベントの通知を Amazon SQS FIFO

キューに送信するには、Amazon EventBridge を使用できます。詳細については、「Amazon EventBridge を有効にします」を参照してください。

トピック

- Amazon S3 コンソールを使用して Amazon SNS、Amazon SQS、または Lambda 通知を有効にします。

Amazon S3 コンソールを使用して Amazon SNS、Amazon SQS、または Lambda 通知を有効にします。

S3 バケットのイベント通知を有効化および設定するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets (バケット)] リストで、イベントを有効にするバケットの名前を選択します。
3. [プロパティ] を選択します。
4. [Event Notifications (イベント通知)] セクションに移動し、[Create event notification (イベント通知の作成)] を選択します。
5. [General configuration (全般設定)] セクションで、イベント通知にわかりやすいイベント名を指定します。オプションで、プレフィックスとサフィックスを指定して、指定した文字で終わるキーを持つオブジェクトのみに通知を制限することもできます。

   a. [Event name (イベント名)] の説明を入力します。

      名前を入力しない場合は、グローバル一意識別子 (GUID) が生成され、名前に使用されます。

   b. (オプション) プレフィックスでイベント通知をフィルタリングするには、プレフィックスを入力します。

      例えば、特定のフォルダ (`images/` など) にファイルが追加されたときにのみ通知を受信するようにプレフィックスフィルタを設定できます。

   c. (オプション) サフィックスでイベント通知をフィルタリングするには、サフィックスを入力します。

      詳細については、「オブジェクトキー名のフィルタリングを使用したイベント通知の設定」を参照してください。

6. イベントタイプセクションで、通知を受信する 1 つ以上のイベントタイプを選択します。

   異なるイベントタイプのリスト化については、SQS、SNS、および Lambda でサポートされているイベントタイプ を参照してください。

7. [Destination (ターゲット)] セクションで、イベント通知の送信先を選択します。

   > **Note**
   >
   > イベント通知を公開する前に、Amazon S3 プリンシパルが関連 API を呼び出すために必要なアクセス許可を付与する必要があります。これは、Lambda 関数、SNS トピック、または SQS キューに通知を発行できるようにするためです。

   a. ターゲットタイプ ([Lambda Function (Lambda 関数)]、[SNS Topic (SNS トピック)]、または [SQS Queue (SQS キュー)]) を選択します。

   b. ターゲットタイプを選択したら、リストから関数、トピック、またはキューを選択します。

   c. または、Amazon リソースネーム (ARN) を指定する場合は、[Enter ARN (ARN を入力)] を選択して ARN を入力します。

   詳細については、「サポートされているイベントの送信先」を参照してください。

8. [Save Changes (変更の保存)] を選択すると、Amazon S3 がテストメッセージをイベント通知の送信先に送信します。

プログラムによるイベント通知の設定

デフォルトで、通知はどのタイプのイベントにも有効ではありません。したがって、通知サブリソースに最初は空の設定で格納されています。

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</NotificationConfiguration>
```

特定のタイプのイベントに対して通知を有効にするには、XML を、Amazon S3 がパブリッシュするイベントのタイプとパブリッシュ先を識別する適切な設定に置き換えます。各宛先には、対応する XML 設定を追加します。

イベントメッセージを SQS キューに発行するには

1 つ以上のイベントタイプの通知の送信先として SQS キューを設定するには、QueueConfiguration を追加します。

```
<NotificationConfiguration>
  <QueueConfiguration>
    <Id>optional-id-string</Id>
    <Queue>sqs-queue-arn</Queue>
    <Event>event-type</Event>
    <Event>event-type</Event>
     ...
  </QueueConfiguration>
   ...
</NotificationConfiguration>
```

イベントメッセージを SNS トピックに発行するには

特定のイベントタイプの通知の送信先として SNS トピックを設定するには、TopicConfiguration を追加します。

```
<NotificationConfiguration>
  <TopicConfiguration>
     <Id>optional-id-string</Id>
     <Topic>sns-topic-arn</Topic>
     <Event>event-type</Event>
     <Event>event-type</Event>
      ...
  </TopicConfiguration>
   ...
</NotificationConfiguration>
```

AWS Lambda 関数を呼び出し、イベントメッセージを引数として指定するには

特定のイベントタイプの通知の送信先として Lambda 関数を設定するには、CloudFunctionConfiguration を追加します。

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>optional-id-string</Id>
     <CloudFunction>cloud-function-arn</CloudFunction>
     <Event>event-type</Event>
     <Event>event-type</Event>
      ...
```

```
  </CloudFunctionConfiguration>
   ...
</NotificationConfiguration>
```

バケットで設定されているすべての通知を削除するには

バケットに設定されたすべての通知を削除するには、notification サブリソースに空の <NotificationConfiguration/> エレメントを保存します。

Amazon S3 は、特定のタイプのイベントを検出すると、そのイベント情報を含むメッセージを発行します。詳細については、「イベントメッセージの構造」を参照してください。

イベント通知の設定の詳細については、次のトピックを参照してください。

- チュートリアル: 通知 (SNS トピックまたは SQS キュー) のバケットを設定する.
- オブジェクトキー名のフィルタリングを使用したイベント通知の設定

チュートリアル: 通知 (SNS トピックまたは SQS キュー) のバケットを設定する

Amazon S3 通知は、Amazon Simple Notification Service (Amazon SNS) または Amazon Simple Queue Service (Amazon SQS) を使用して受信できます。このチュートリアルでは、Amazon SNS トピックと Amazon SQS キューを使用して、バケットに通知設定を追加します。

> **Note**
>
> Amazon Simple キューサービス FIFO (先入れ先出し) キューは、Amazon S3 イベント通知の送信先としてサポートされていません。Amazon S3 イベントの通知を Amazon SQS FIFO キューに送信するには、Amazon EventBridge を使用できます。詳細については、「Amazon EventBridge を有効にします」を参照してください。

トピック

- チュートリアルの概要
- ステップ 1: Amazon SQS キューを作成する
- ステップ 2: Amazon SNS トピックを作成する
- ステップ 3: 通知設定をバケットに追加する
- ステップ 4: セットアップをテストする

チュートリアルの概要

このチュートリアルは、以下を行う際に役立ちます。

- `s3:ObjectCreated:*` タイプのイベントを Amazon SQS キューに発行する。
- `s3:ReducedRedundancyLostObject` タイプのイベントを Amazon SNS トピックに発行する。

通知設定の情報については、Amazon SQS、Amazon SNS、Lambda を使用します を参照してください。

コンソールを使用して、コードを記述することなく、これらの手順をすべて実行できます。また、通知設定をプログラムで追加できるように、AWS SDK for Java および .NET を使用したコード例も用意されています。

この手順には、以下のステップが含まれます。

1. Amazon SQS キュー を作成する。

   Amazon SQS コンソールを使用して、SQS キューを作成します。Amazon S3 がそのキューにプログラムで送信するどのメッセージにもアクセスできます。ただし、このチュートリアルでは、コンソールで通知メッセージを確認します。

   キューにアクセスポリシーをアタッチして、メッセージを発行するための Amazon S3 アクセス許可を付与します。
2. Amazon SNS トピックを作成する。

   Amazon SNS コンソールを使用して、SNS トピックを作成し、トピックにサブスクライブします。これにより、そこに投稿されたすべてのイベントが配信されます。通信プロトコルとして E メールを指定します。トピックを作成すると、Amazon SNS から E メールが送信されます。E メール内のリンクを使用して、トピックのサブスクリプションを確認します。

   トピックにアクセスポリシーをアタッチして、メッセージを発行するための Amazon S3 アクセス許可を付与します。
3. 通知設定をバケットに追加します。

ステップ 1: Amazon SQS キューを作成する

次の手順に従い、Amazon Simple Queue Service (Amazon SQS) キューを作成し、このキューにサブスクライブします。

1. Amazon SQS コンソールを使用して、キューを作成します。手順については、「Amazon Simple Queue Service デベロッパーガイド」の「Getting Started with Amazon SQS」を参照してください。
2. キューに添付したアクセスポリシーを次のポリシーに置き換えます。

   a. Amazon SQS コンソールの [Queues] (キュー) リストで、キューの名前を選択します。

   b. [Access policy] (アクセスポリシー) タブで [Edit] (編集) をクリックします。

   c. キューに添付されているアクセスポリシーを置き換えます。その中で、Amazon SQS ARN、ソースバケット名、バケット所有者アカウント ID を提供します。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SQS:SendMessage"
            ],
            "Resource": "SQS-queue-ARN",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

   d. [Save] を選択します。

3. (オプション) Amazon SQS キューまたは Amazon SNS トピックで、AWS Key Management Service (AWS KMS) によるサーバー側の暗号化が有効になっている場合は、次のポリシーを関連する対称暗号化カスタマーマネージドキーに追加します。

Amazon SQS または Amazon SNS 用の AWS マネージドキーは変更できないため、カスタマーマネージドキーにポリシーを追加する必要があります。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS KMS で Amazon SQS および Amazon SNS で SSE を使用する方法の詳細については、以下を参照してください。

- 「Amazon Simple Notification Service デベロッパーガイド」の「キー管理」。
- 「Amazon Simple Queue Service デベロッパーガイド」の「キー管理」。

4. キューの ARN を書き留めます。

作成した SQS キューは、AWS アカウント 内の別のリソースです。固有の Amazon リソースネーム (ARN) があります。この ARN は次のステップで必要になります。ARN の形式は次のとおりです。

```
arn:aws:sqs:aws-region:account-id:queue-name
```

ステップ 2: Amazon SNS トピックを作成する

手順に従って、Amazon SNS トピックを作成してサブスクライブします。

1. Amazon SNS コンソールを使用してトピックを作成します。詳細については、「Amazon Simple Notification Service デベロッパーガイド」の「Creating an Amazon SNS topic」を参照してください。
2. トピックを受信します。この演習では、通信プロトコルとしてメールを使用します。手順については、Amazon Simple Notification Service デベロッパーガイドの Amazon SNS トピックへのサブスクライブを参照してください。

   トピックのサブスクリプションの確認を求めるメールがお客様宛てに送信されます。サブスクリプションを確認します。
3. トピックにアタッチされたアクセスポリシーを次のポリシーに置き換えます。その中で、SNS トピックの ARN、バケット名、バケット所有者のアカウント ID を提供します。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "Example SNS topic policy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "SNS-topic-ARN",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:bucket-name"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

4. トピックの ARN を書き留めておきます。

作成した SNS トピックは、AWS アカウント内の別のリソースであり、一意の ARN があります。この ARN は次のステップで必要になります。ARN は次のような形式になります。

```
arn:aws:sns:aws-region:account-id:topic-name
```

ステップ 3: 通知設定をバケットに追加する

Amazon S3 コンソールを使用するか、プログラムで AWS SDK を使用して、バケットの通知を有効にすることができます。バケットでの通知を設定するオプションのいずれかを選択します。このセクションでは、AWS SDK for Java および .NET を使用するコード例を示します。

オプション A: コンソールを使用してバケットの通知を有効にする

Amazon S3 コンソールを使用して、Amazon S3 に次のアクションをリクエストする通知設定を追加します。

- [All object create events (すべてのオブジェクト作成イベント)] タイプのイベントを Amazon SQS キューに発行する。
- [Object in RRS lost (RRS オブジェクトの紛失)] タイプのイベントを Amazon SNS トピックに発行する。

通知設定を保存すると、Amazon S3 からテストメッセージが発行され、このメッセージが E メールで送信されます。

手順については、Amazon S3 コンソールを使用したイベント通知の有効化と設定 を参照してください。

オプション B: AWS SDK を使用してバケットの通知を有効にする

.NET

次の C# のコード例では、バケットに通知設定を追加する完全なコードのリストを示します。コードを更新して、バケット名と SNS トピックの ARN を提供する必要があります。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
```

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class EnableNotificationsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string snsTopic = "*** SNS topic ARN ***";
        private const string sqsQueue = "*** SQS topic ARN ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            EnableNotificationAsync().Wait();
        }

        static async Task EnableNotificationAsync()
        {
            try
            {
               PutBucketNotificationRequest request = new
 PutBucketNotificationRequest
                {
                    BucketName = bucketName
                };

                TopicConfiguration c = new TopicConfiguration
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic
                };
                request.TopicConfigurations = new List<TopicConfiguration>();
                request.TopicConfigurations.Add(c);
                request.QueueConfigurations = new List<QueueConfiguration>();
                request.QueueConfigurations.Add(new QueueConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedPut },
                    Queue = sqsQueue
```

```
                });

                PutBucketNotificationResponse response = await
 client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' ",
 e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown error encountered on server.
 Message:'{0}' ", e.Message);
            }
        }
    }
}
```

Java

次の例では、バケットに通知設定を追加する方法を示します。作業サンプルの作成方法およびテスト方法については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.EnumSet;

public class EnableNotificationOnABucket {

    public static void main(String[] args) throws IOException {
        String bucketName = "*** Bucket name ***";
        Regions clientRegion = Regions.DEFAULT_REGION;
        String snsTopicARN = "*** SNS Topic ARN ***";
```

```
        String sqsQueueARN = "*** SQS Queue ARN ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();
            BucketNotificationConfiguration notificationConfiguration = new
 BucketNotificationConfiguration();

            // Add an SNS topic notification.
            notificationConfiguration.addConfiguration("snsTopicConfig",
                    new TopicConfiguration(snsTopicARN,
 EnumSet.of(S3Event.ObjectCreated)));

            // Add an SQS queue notification.
            notificationConfiguration.addConfiguration("sqsQueueConfig",
                    new QueueConfiguration(sqsQueueARN,
 EnumSet.of(S3Event.ObjectCreated)));

            // Create the notification configuration request and set the bucket
 notification
            // configuration.
            SetBucketNotificationConfigurationRequest request = new
 SetBucketNotificationConfigurationRequest(
                    bucketName, notificationConfiguration);
            s3Client.setBucketNotificationConfiguration(request);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

ステップ 4: セットアップをテストする

さて、バケットにオブジェクトをアップロードしたり、Amazon SQS コンソールのイベント通知を検証することでセットアップをテストできます。手順については、Amazon Simple Queue Service デベロッパーガイドの開始方法」セクションのメッセージの受信を参照してください。

オブジェクトキー名のフィルタリングを使用したイベント通知の設定

Amazon S3 イベント通知を設定する場合、Amazon S3 による通知の送信をトリガーするサポートターゲットの Amazon S3 イベントタイプを指定する必要があります。指定しなかったイベントタイプが S3 バケットで発生しても、Amazon S3 は通知を送信しません。

通知がオブジェクトのキー名のプレフィックスまたはサフィックスでフィルタリングされるように設定できます。例えば、ファイル名の拡張子が `.jpg` であるイメージファイルがバケットに追加されたときにのみ、通知が送信されるように設定できます。または、「`images/`」というプレフィックスの付いたオブジェクトがバケットに追加されたときに Amazon SNS トピックに通知が送信されるように設定し、同じバケット内にある「`logs/`」というプレフィックスの付いたオブジェクトの通知が AWS Lambda 関数に渡されるようにすることができます。

> **Note**
>
> ワイルドカード文字 (「*」) は、フィルタでプレフィックスまたはサフィックスとして使用することはできません。プレフィックスまたはサフィックスにスペースが含まれている場合は、それを「+」文字に置き換える必要があります。プレフィックスまたはサフィックスの値に他の特殊文字を使用する場合は、URL エンコード (パーセントエンコード) 形式で入力する必要があります。イベント通知のプレフィックスまたはサフィックスに使用する場合で、URL エンコード形式に変換する必要がある特殊文字の一覧については、「セーフ文字」を参照してください。

Amazon S3 コンソールでオブジェクトキー名フィルタリングを使用する通知設定を設定できます。これを行うには、AWS SDK または REST API を介して、Amazon S3 API を直接使用します。コンソール UI を使用してバケットの通知設定をセットアップする方法については、Amazon S3 コンソールを使用したイベント通知の有効化と設定 を参照してください。

Amazon S3 は、「Amazon SQS、Amazon SNS、Lambda を使用します」で説明されているように、バケットに関連付けられている notification サブリソースに通知設定を XML 形式で保存します。`Filter` XML 構造を使用して、オブジェクトキー名のプレフィックスまたはサフィックスで

フィルタして通知のルールを定義します。Filter XML 構造の詳細については、Amazon Simple Storage Service API リファレンスの PUT Bucket 通知 を参照してください。

Filter を使用する通知設定では、プレフィックスの重複、サフィックスの重複、またはプレフィックスとサフィックスの重複があるフィルタリングルールを定義できません。以下のセクションでは、オブジェクトキー名フィルタリングを使用した有効な通知設定の例を示します。プレフィックス/サフィックスが重複しているために無効である通知設定の例も示します。

トピック

- オブジェクトキー名によるフィルタ処理を使用した有効な通知設定の例
- 無効なプレフィックスやサフィックスの重複がある通知設定の例。

オブジェクトキー名によるフィルタ処理を使用した有効な通知設定の例

次の通知設定には、Amazon SQS キューを識別するキュー設定が含まれており、これにより Amazon S3 は s3:ObjectCreated:Put タイプのイベントを発行できます。イベントは、images/ のプレフィックスと jpg サフィックスの付いたオブジェクトがバケットに PUT されるたびに発行されます。

```
<NotificationConfiguration>
  <QueueConfiguration>
      <Id>1</Id>
      <Filter>
          <S3Key>
              <FilterRule>
                  <Name>prefix</Name>
                  <Value>images/</Value>
              </FilterRule>
              <FilterRule>
                  <Name>suffix</Name>
                  <Value>jpg</Value>
              </FilterRule>
          </S3Key>
      </Filter>
      <Queue>arn:aws:sqs:us-west-2:444455556666:s3notificationqueue</Queue>
      <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
</NotificationConfiguration>
```

次の通知設定には、複数の重複していないプレフィックスがあります。この設定では、images/ フォルダ内の PUT リクエストの通知を queue-A に送信し、logs/ フォルダ内の PUT リクエストの通知を queue-B に送信するように定義しています。

```
<NotificationConfiguration>
  <QueueConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images/</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-A</Queue>
     <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
  <QueueConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>logs/</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-B</Queue>
     <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
</NotificationConfiguration>
```

次の通知設定には、複数の重複していないサフィックスがあります。この設定では、バケットに新しく追加される .jpg イメージは Lambda の cloud-function-A によって処理され、新しく追加される .png イメージは cloud-function-B によって処理されるように定義されています。.png と .jpg は、末尾の 1 文字が同じであっても、重複していると見なされません。文字列が両方のサフィックスで終わる可能性がある場合には、2 つのサフィックスは、重複していると見なされます。文字列が .png と .jpg の両方で終わることはないので、この設定例の 2 つのサフィックスは重複していません。

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.jpg</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.png</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

Filter を使用する通知設定では、同じイベントタイプのプレフィックスが重複するフィルタリングルールを定義できません。重複しないサフィックスと一緒に使用される重複するプレフィックスがある場合にのみ、そうすることができます。次の設定例は、プレフィックスは重複しており、サフィックスは重複していないオブジェクトがどのように別々の場所に送られるかを示しています。

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
```

```
                    <Name>prefix</Name>
                    <Value>images</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.jpg</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.png</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

無効なプレフィックスやサフィックスの重複がある通知設定の例。

ほとんどの場合、Filter を使用する通知構成では、同じイベントタイプのプレフィックス、サフィックス、またはプレフィックスとサフィックスの組み合わせが重複するフィルタのルールを定義できません。サフィックスが重複していなければ、プレフィックスが重複していても問題ありません。例については、オブジェクトキー名のフィルタリングを使用したイベント通知の設定 を参照してください。

イベントタイプが異なれば、重複しているオブジェクトキー名フィルタを使用できます。例えば、image/ イベントタイプにプレフィックス ObjectCreated:Put と image/ イベントタイプにプレフィックス ObjectRemoved:* を使用する通知設定を作成できます。

Amazon S3 コンソールまたは API の使用時に、同じイベントタイプに対して名前が重複する無効なフィルタを使用する通知設定を保存しようとすると、エラーが発生します。このセクションでは、重複する名前のフィルタにより、無効である通知設定の例を示します。

既存の通知構成ルールには、他のプレフィックスとサフィックスとそれぞれ一致するデフォルトのプレフィックスとサフィックスがあると想定されます。次の通知設定は、重複するプレフィックスがあるために無効です。具体的には、ルートプレフィックスが他のプレフィックスと重複します。この例のように、プレフィックスではなくサフィックスを使用するのも true です。ルートサフィックスが他のサフィックスと重複します。

```
<NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-two</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>prefix</Name>
                     <Value>images</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration>
```

次の通知設定は、重複するサフィックスがあるために無効です。文字列が両方のサフィックスで終わる可能性がある場合には、2 つのサフィックスは、重複していると見なされます。文字列は jpg とpg で終わることができます。したがって、サフィックスは重なり合います。プレフィックスについても同様です。文字列が両方のプレフィックスで始まる可能性があれば、2 つのプレフィックスは重複していると見なされます。

```
 <NotificationConfiguration>
     <TopicConfiguration>
```

```
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
         <Event>s3:ObjectCreated:Put</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>pg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration
```

次の通知設定は、重複するプレフィックスとサフィックスがあるために無効です。

```
<NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>prefix</Name>
                     <Value>images</Value>
                 </FilterRule>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
```

```
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
         <Event>s3:ObjectCreated:Put</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration>
```

イベントメッセージの構造

Amazon S3 がイベントを発行するために送信する通知メッセージは JSON 形式です。

イベント通知の設定に関する一般的な概要と手順については、Amazon S3 イベント通知 を参照してください。

次の例は、イベント通知 JSON 構造体のバージョン 2.2 を示します。Amazon S3 は、このイベント構造体のバージョン 2.1 、 2.2 、 2.3 を使用します。Amazon S3 は、クロスリージョンレプリケーションイベント通知にバージョン 2.2 を使用します。S3 ライフサイクル、S3 Intelligent-Tiering、オブジェクト ACL、オブジェクトのタグ付け、オブジェクトの復元削除イベントにバージョン 2.3 を使用します。これらのバージョンには、これらのオペレーションに固有の追加情報が含まれています。バージョン 2.2 および 2.3 は、バージョン 2.1 と互換性があり、Amazon S3 が現在すべてのイベント通知タイプで使用されています。

```
{
   "Records":[
      {
         "eventVersion":"2.2",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"The time, in ISO-8601 format, for example,
 1970-01-01T00:00:00.000Z, when Amazon S3 finished processing the request",
         "eventName":"event-type",
         "userIdentity":{
            "principalId":"Amazon-customer-ID-of-the-user-who-caused-the-event"
         },
```

```
         "requestParameters":{
            "sourceIPAddress":"ip-address-where-request-came-from"
         },
         "responseElements":{
            "x-amz-request-id":"Amazon S3 generated request ID",
            "x-amz-id-2":"Amazon S3 host that processed the request"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"ID found in the bucket notification configuration",
            "bucket":{
               "name":"bucket-name",
               "ownerIdentity":{
                  "principalId":"Amazon-customer-ID-of-the-bucket-owner"
               },
               "arn":"bucket-ARN"
            },
            "object":{
               "key":"object-key",
               "size":"object-size in bytes",
               "eTag":"object eTag",
               "versionId":"object version if bucket is versioning-enabled, otherwise
 null",
               "sequencer": "a string representation of a hexadecimal value used to
 determine event sequence, only used with PUTs and DELETEs"
            }
         },
         "glacierEventData": {
            "restoreEventData": {
               "lifecycleRestorationExpiryTime": "The time, in ISO-8601 format, for
 example, 1970-01-01T00:00:00.000Z, of Restore Expiry",
               "lifecycleRestoreStorageClass": "Source storage class for restore"
            }
         }
      }
   ]
}
```

イベントメッセージ構造について、以下の点に注意してください。

- <major> 形式でメジャーおよびマイナーバージョンを含む eventVersion キーの値。<minor>

Amazon S3 が、後方互換性のないイベント構造に変更を加えた場合、メジャーバージョンは増分されます。これには、すでに存在する JSON フィールドの削除や、フィールドのコンテンツの表現方法 (日付形式など) の変更が含まれます。

Amazon S3 が、イベント構造に新しいフィールドを追加した場合、マイナーバージョンは増分されます。これは、既存のイベントの一部またはすべてに対して新しい情報が提供された場合に発生します。これは、新しく導入されたイベントタイプでのみ新しい情報が提供されている場合にも発生します。イベント構造の新しいマイナーバージョンとの将来の互換性を保つには、アプリケーションは新しいフィールドを無視する必要があります。

新しいイベントタイプが導入したが、イベントの構造が変更されていない場合、イベントのバージョンは変更されません。

アプリケーションがイベント構造を正しく解析できるようにするため、メジャーバージョン番号が同等かどうかの比較を行うことをお勧めします。アプリケーションで想定されるフィールドが存在することを確認するため、マイナーバージョンが同等以上かを比較することをお勧めします。

- `eventName` はイベント通知タイプのリストを参照しますが、`s3:` プレフィックスが含まれていません。
- `responseElements` キー値は、AWS Support のサポートによってリクエストを追跡する場合に役立ちます。`x-amz-request-id` と `x-amz-id-2` のどちらも、Amazon S3 が個々のリクエストを追跡するのに役立ちます。これらの値は、イベントを開始するリクエストへの応答として Amazon S3 が返す値と同じです。これは、イベントをリクエストに一致させるために使用できます。
- `s3` キーは、イベントに関与したバケットとオブジェクトに関する情報を提供します。オブジェクトのキー名の値は URL エンコードされます。例えば、「red flower.jpg」は「red+flower.jpg」となります (Amazon S3 はコンテンツタイプとして「`application/x-www-form-urlencoded`」をレスポンスで返します)。
- イベントのシーケンスを決定する方法の 1 つとして、`sequencer` キーがあります。イベントが発生した順序でイベント通知が届く保証はありません。ただし、オブジェクト (PUT) を作成するイベントからの通知 と削除オブジェクトは `sequencer` を含みます。これは、特定のオブジェクトキーのイベントの順序を決定するために使用できます。

  同じオブジェクトキーに対する 2 つのイベント通知の `sequencer` の文字列を比較すると、`sequencer` の 16 進値が大きいほうのイベント通知が後に発生したイベントであることがわかります。イベント通知を使用して Amazon S3 オブジェクトの別のデータベースまたはインデッ

クスを維持している場合は、イベント通知を処理するたびに sequencer の値を比較し、保存することを推奨します。

次の点に注意してください。

- 複数のオブジェクトキーのイベントの順序を決定するために sequencer を使用することはできません。
- sequencer の長さが異なる場合があります。これらの値を比較するには、最初に短い方の値を右に 0 と挿入してから、辞書式比較を実行します。

- glacierEventData キーは s3:ObjectRestore:Completed イベントに対してのみ表示されます。
- restoreEventData キーは、復元リクエストに関連する属性が含まれます。
- replicationEventData キーは、レプリケーションイベントに対してのみ表示されます。
- intelligentTieringEventData キーは S3 Intelligent-Tiering イベントのみ表示されます。
- lifecycleEventData キーは S3 ライフサイクルの移行イベントのみ表示されます。

メッセージの例

Amazon S3 イベント通知メッセージの例を次に示します。

Amazon S3 テストメッセージ

バケットにイベント通知を設定すると、Amazon S3 は次のようなテストメッセージを送信します。

```
{
   "Service":"Amazon S3",
   "Event":"s3:TestEvent",
   "Time":"2014-10-13T15:57:02.089Z",
   "Bucket":"bucketname",
   "RequestId":"5582815E1AEA5ADF",
   "HostId":"8cLeGAmw098X5cv4Zkwcmo8vvZa3eH3eKxsPzbB9wrR+YstdA6Knx4Ip8EXAMPLE"
}
```

PUT リクエストを使用してオブジェクトが作成された場合のメッセージの例

次のメッセージは、Amazon S3 が s3:ObjectCreated:Put イベントを発行するために送信するメッセージの例です。

```
{
```

```
   "Records":[
      {
         "eventVersion":"2.1",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"ObjectCreated:Put",
         "userIdentity":{
            "principalId":"AIDAJDPLRKLG7UEXAMPLE"
         },
         "requestParameters":{
            "sourceIPAddress":"127.0.0.1"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"mybucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::mybucket"
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
               "versionId":"096fKKXTRTtl3on89fVO.nfljtsv6qko",
               "sequencer":"0055AED6DCD90281E5"
            }
         }
      }
   ]
}
```

各 IAM 識別プレフィクス (AIDA、AROA、AGPA など) の定義については、IAM ユーザーガイドの IAM 識別子を参照してください。

EventBridge の使用

Amazon S3 は、バケット内で特定のイベントが発生するたびに Amazon EventBridge にイベントを送信できます。他の宛先とは異なり、配信するイベントタイプを選択する必要はありません。EventBridge を有効にすると、以下のすべてのイベントが EventBridge に送信されます。EventBridge ルールを使用すると、イベントを追加のターゲットにルートできます。Amazon S3 が EventBridge に送信するイベントを以下に示します。

イベントタイプ	説明
オブジェクトの作成	オブジェクトが作成されました。 イベントメッセージ構造の理由フィールドは、PutObject、POST Object、CopyObject、CompleteMultipartUpload などのオブジェクトの作成に使用された S3 API を示します。
オブジェクトが削除されました (DeleteObject) オブジェクトが削除されました (ライフサイクルの有効期限切れ)	オブジェクトが削除されました。 S3 API コールを使用してオブジェクトを削除すると、理由フィールドが deleteObject に設定されます。S3 ライフサイクルの有効期限切れルールによってオブジェクトを削除すると、理由フィールドがライフサイクルの有効期限切れに設定されます。詳細については、「オブジェクトの有効期限」を参照してください。 バージョン管理されていないオブジェクトが削除されるか、バージョン管理されたオブジェクトが完全に削除されると、削除タイプフィールドは PermanentlyDeleted に設定されます。バージョン管理されたオブジェクトに対して削除マーカーが作成されると、削除タイプフィールドは作成したマーカーの削除に設定されます。詳細については、「バージョニングが有効なバケットからのオブジェクトバージョンの削除」を参照してください。
オブジェクトの復元の開始	オブジェクトの復元は、S3 Glacier、S3 Glacier Deep Archive ストレージクラス、S3 Intelligent-Tiering アーカイブアクセス、Deep Archive アクセス階層から開始されまし

イベントタイプ	説明
	た。詳細については、「アーカイブされたオブジェクトの操作」を参照してください。
オブジェクトの復元の完了	オブジェクトの復元が完了しました。
オブジェクトの復元の期限切れ	S3 Glacier または S3 Glacier Deep Archive から復元されたオブジェクトの一時コピーの有効期限切れのため削除されました。
オブジェクトストレージクラスの変更	オブジェクトが別のストレージクラスに移行されました。詳細については、「Amazon S3 ライフサイクルを使用したオブジェクトの移行」を参照してください。
オブジェクトアクセス階層が変更されました	オブジェクトが S3 Intelligent-Tiering アーカイブアクセス階層または Deep Archive アクセス階層に移行されました。詳細については、「Amazon S3 Intelligent-Tiering」を参照してください。
オブジェクト ACL が更新されました	オブジェクトのアクセスコントロールリスト (ACL) が PutObjectACL を使用して設定されました。リクエストによってオブジェクトの ACL が変更されない場合、イベントは生成されません。詳細については、「アクセスコントロールリスト (ACL) の概要」を参照してください。
追加されたオブジェクトタグ	PutObjectTagging を使用して、一連のタグがオブジェクトに追加されました。詳細については、「タグを使用してストレージを分類する」を参照してください。
削除済みのオブジェクトタグ	DeleteObjectTagging を使用して、すべてのタグがオブジェクトから削除されました。詳細については、「タグを使用してストレージを分類する」を参照してください。

> **Note**
>
> Amazon S3 イベントタイプを EventBridge イベントタイプにマッピングする方法の詳細については、Amazon EventBridge のマッピングとトラブルシューティング を参照してください。

EventBridge で Amazon S3 イベント通知を使用して、バケットでイベントが発生したときにアクションを実行するルールを記述できます。例えば、通知を送信するように設定できます。詳細については、Amazon EventBridge ユーザーガイドの Amazon EventBridge とは を参照してください。

EventBridge API を使用して利用できるアクションとデータ型の詳細については、「Amazon EventBridge API リファレンス」の「Amazon EventBridge API Reference」を参照してください。

料金の詳細については、Amazon EventBridge の料金を参照してください。

トピック

Amazon EventBridge アクセス許可

Amazon S3 では、Amazon EventBridge にイベントを配信するための追加のアクセス権限は必要ありません。

Amazon EventBridge を有効にします

S3 コンソール AWS Command Line Interface (AWS CLI)、または Amazon S3 REST API を使用して Amazon EventBridge を有効にできます。

S3 コンソールの使用

S3 コンソールで EventBridge イベント配信を有効にします。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

2. [Buckets (バケット)] リストで、イベントを有効にするバケットの名前を選択します。
3. [プロパティ] を選択します。
4. イベント通知セクションに移動し、Amazon EventBridge サブセクションを見つけます。[編集] を選択します。
5. このバケット内のすべてのイベント用の Amazon EventBridge に通知を送信するの下にある On を選択します。

> **Note**
>
> EventBridge を有効にすると、変更が適用されるまで約 5 分かかります。

AWS CLI の使用

次の例では、DOC-EXAMPLE-BUCKET1 Amazon EventBridge が有効であるバケットのバケット通知設定を作成します。

```
aws s3api put-bucket-notification-configuration --bucket example-s3-bucket1 --
notification-configuration='{ "EventBridgeConfiguration": {} }'
```

REST API の使用

Amazon S3 REST API を呼び出して、バケット上で Amazon EventBridge をプログラムで有効にできます。詳細については、Amazon S3 API リファレンスの PutBucketNotificationConfiguration を参照してください。

次の例は、Amazon EventBridge を有効にしたバケット通知設定を作成するために使用される XML を示しています。

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <EventBridgeConfiguration>
  </EventBridgeConfiguration>
</NotificationConfiguration>
```

EventBridge ルールの作成

有効にすると、特定のタスクに対して Amazon EventBridge ルールを作成できます。例えば、オブジェクトが作成されたときに電子メール通知を送信できます。完全なチュートリアルについて

は、Amazon EventBridge ユーザーガイドのチュートリアル: Amazon S3 オブジェクトが作成されたとき通知を送信するを参照してください。

EventBridge イベントメッセージの構造

Amazon S3 がイベントを発行するために送信する通知メッセージは JSON 形式です。Amazon S3 が Amazon EventBridge にイベントを送信すると、次のフィールドが表示されます。

- バージョン — 現在、すべてのイベントで 0 (ゼロ)。
- id – イベントごとに生成されるバージョン 4 UUID。
- 詳細タイプ — 送信されるイベントのタイプ。イベントタイプのリスト化については、EventBridge の使用 を参照してください。
- source – イベントを生成したサービスを識別します。
- アカウント — バケット所有者の 12 桁の AWS アカウント ID。
- 時間 — イベントが発生した時刻。
- リージョン — バケットの AWS リージョン を識別します。
- リソース — バケットの Amazon リソースネーム (ARN) を含む JSON 配列。
- 詳細 – イベントに関する情報を含む JSON オブジェクト。このフィールドに含めることができる内容の詳細については、「イベントメッセージ詳細フィールド」を参照してください。

イベントメッセージの構造の例

Amazon EventBridge に送信できる Amazon S3 イベント通知メッセージの例を次に示します。

オブジェクトの作成

```
{
  "version": "0",
  "id": "17793124-05d4-b198-2fde-7ededc63b103",
  "detail-type": "Object Created",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
  ],
  "detail": {
```

```
    "version": "0",
    "bucket": {
      "name": "DOC-EXAMPLE-BUCKET1"
    },
    "object": {
      "key": "example-key",
      "size": 5,
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "IYV3p45BT0ac8hjHg1houSdS1a.Mro8e",
      "sequencer": "617f08299329d189"
    },
    "request-id": "N4N7GDK58NMKJ12R",
    "requester": "123456789012",
    "source-ip-address": "1.2.3.4",
    "reason": "PutObject"
  }
}
```

(DeleteObject を使用して) オブジェクトが削除されました

```
{
  "version": "0",
  "id": "2ee9cc15-d022-99ea-1fb8-1b1bac4850f9",
  "detail-type": "Object Deleted",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "DOC-EXAMPLE-BUCKET1"
    },
    "object": {
      "key": "example-key",
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "1QW9g1Z99LUNbvaaYVpW9xDlOLU.qxgF",
      "sequencer": "617f0837b476e463"
    },
```

```
    "request-id": "0BH729840619AG5K",
    "requester": "123456789012",
    "source-ip-address": "1.2.3.4",
    "reason": "DeleteObject",
    "deletion-type": "Delete Marker Created"
  }
}
```

(ライフサイクルの有効期限切れのため) オブジェクトが削除されました

```
{
  "version": "0",
  "id": "ad1de317-e409-eba2-9552-30113f8d88e3",
  "detail-type": "Object Deleted",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "DOC-EXAMPLE-BUCKET1"
    },
    "object": {
      "key": "example-key",
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "mtB0cV.jejK63XkRNceanNMC.qXPWLeK",
      "sequencer": "617b398000000000"
    },
    "request-id": "20EB74C14654DC47",
    "requester": "s3.amazonaws.com",
    "reason": "Lifecycle Expiration",
    "deletion-type": "Delete Marker Created"
  }
}
```

オブジェクトの復元の完了

```
{
  "version": "0",
  "id": "6924de0d-13e2-6bbf-c0c1-b903b753565e",
  "detail-type": "Object Restore Completed",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "DOC-EXAMPLE-BUCKET1"
    },
    "object": {
      "key": "example-key",
      "size": 5,
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "KKsjUC1.6gIjqtvhfg5AdMI0eCePIiT3"
    },
    "request-id": "189F19CB7FB1B6A4",
    "requester": "s3.amazonaws.com",
    "restore-expiry-time": "2021-11-13T00:00:00Z",
    "source-storage-class": "GLACIER"
  }
}
```

イベントメッセージ詳細フィールド

詳細フィールドは、イベントに関する情報のある JSON オブジェクトを含みます。詳細フィールドには、次のフィールドが存在する可能性があります。

- バージョン — 現在、すべてのイベントで 0 (ゼロ)。
- バケット — イベントに関与した Amazon S3 バケットに関する情報。
- ターゲット — イベントに関与した Amazon S3 オブジェクトに関する情報。
- リクエスト ID — S3 レスポンスのリクエスト ID。

- リクエスタ — AWS アカウント ID または AWS リクエスタのサービスプリンシパル。
- 送信元 IP アドレス — S3 リクエストの出典 IP アドレス。S3 リクエストによってトリガーされたイベントにのみ存在します。
- 理由 — オブジェクトの作成イベントの場合、オブジェクトの作成に使用される S3 API: PutObject、POST Object、CopyObject、CompleteMultipartUpload。オブジェクトの作成イベントの場合、これは、オブジェクトが S3 API コールによって削除された場合は DeleteObject に設定され、オブジェクトが S3 ライフサイクル有効期限切れルールによって削除された場合は LifecycleExpiration に設定されます。詳細については、「オブジェクトの有効期限」を参照してください。
- deletion-type (削除タイプ) — オブジェクトの削除イベントの場合、バージョン管理されていないオブジェクトが削除されるか、バージョン管理されたオブジェクトが完全に削除されると、これは完全に削除されます。バージョン化したオブジェクトに削除マーカーが作成されると、これは削除マーカーの作成に設定されます。詳細については、「バージョニングが有効なバケットからのオブジェクトバージョンの削除」を参照してください。
- restore-expiry-time (保存有効期限) — オブジェクト保存完了 イベントの場合、オブジェクトの一時コピーが S3 から削除される時刻。詳細については、「アーカイブされたオブジェクトの操作」を参照してください。
- source-storage-class (ソースストレージクラス) — オブジェクト復元開始およびオブジェクト復元完了イベントの場合の、復元されるオブジェクトのストレージクラス。詳細については、「アーカイブされたオブジェクトの操作」を参照してください。
- destination-storage-class (デスティネーション・ストレージ・クラス) — オブジェクトストレージクラスの変更イベントの場合の、オブジェクトの新しいストレージクラス。詳細については、「Amazon S3 ライフサイクルを使用したオブジェクトの移行」を参照してください。
- destination-access-tier (宛先アクセス階層) — オブジェクトアクセス階層の変更イベントの場合、オブジェクトの新しいアクセス階層。詳細については、「Amazon S3 Intelligent-Tiering」を参照してください。

Amazon EventBridge のマッピングとトラブルシューティング

次の表では、Amazon S3 イベントタイプが Amazon EventBridge イベントタイプにどのようにマッピングされるかを説明します。

S3 イベントタイプ	Amazon EventBridge の詳細タイプ
ObjectCreated:Put	オブジェクトの作成

S3 イベントタイプ	Amazon EventBridge の詳細タイプ
ObjectCreated:Post ObjectCreated:Copy ObjectCreated:CompleteMultipartUpload	
ObjectRemoved:Delete ObjectRemoved:DeleteMarkerCreated LifecycleExpiration:Delete LifecycleExpiration:DeleteMarkerCreated	削除したオブジェクト
ObjectRestore:Post	オブジェクトの復元の開始
ObjectRestore:Completed	オブジェクト復元の完了
ObjectRestore:Delete	オブジェクト復元の有効期限切れ
LifecycleTransition	オブジェクトストレージクラスの変更
IntelligentTiering	オブジェクトアクセス階層が変更されました
ObjectTagging:Put	追加されたオブジェクトタグ
ObjectTagging:Delete	削除済みのオブジェクトタグ
ObjectACL:Put	オブジェクト ACL が更新されました

Amazon EventBridge のトラブルシューティング

EventBridge のトラブルシューティング方法の詳細については、「Amazon EventBridge ユーザーガイド」の「Troubleshooting Amazon EventBridgeを参照してください 。

分析とインサイトの使用

Amazon S3 の分析とインサイトを使用して、ストレージの使用状況を把握、分析、最適化することができます。詳細については、以下のトピックを参照してください。

トピック

Amazon S3 分析 – ストレージクラス分析

Amazon S3 分析のストレージクラス分析を使用することにより、ストレージアクセスパターンを分析し、適切なデータをいつ適切なストレージクラスに移行すべきかを判断できます。この新しい Amazon S3 分析機能は、アクセス頻度の低い STANDARD ストレージをいつ STANDARD_IA (IA: 小頻度アクセス) ストレージクラスに移行すべきかを判断できるように、データアクセスパターンを確認します。ストレージクラスの詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。

ストレージクラス分析が、フィルタリングされたデータセットの小頻度アクセスパターンを一定期間監視すると、分析結果を使用してライフサイクル設定を改善できます。ストレージクラス分析を設定し、バケット内のすべてのオブジェクトを分析できます。または、共通のプレフィックス (つまり、共通の文字列で始まる名前を持つオブジェクト)、オブジェクトタグ、プレフィックスとタグの両方によってオブジェクトを分析のためにグループ化するフィルターを設定できます。多くの場合、ストレージクラス分析を有効活用するには、オブジェクトグループによるフィルタリングが最適な方法です。

> **Important**
>
> ストレージクラス分析では、標準 ~ 標準 IA クラスのレコメンデーションのみが提供されます。

バケットあたり複数のストレージクラス分析フィルター (最大 1,000 個) を設定でき、フィルターごとに別個の分析を受け取ります。複数のフィルター設定を使用すると、オブジェクトの特定のグループを分析し、オブジェクトを STANDARD_IA に移行するライフサイクル設定を改善できます。

ストレージクラス分析では、Amazon S3 コンソールにストレージの使用状況が可視化され、毎日更新されます。この日次の使用状況データを S3 バケットにエクスポートし、スプレッドシートアプリケーションで、または Amazon QuickSight などのビジネスインテリジェンスツールで表示することもできます。

ストレージクラスの分析に関連するコストがあります。料金については、「管理とレプリケーション」および「Amazon S3 の料金」を参照してください。

トピック

ストレージクラス分析をセットアップする方法

ストレージクラス分析は、分析するオブジェクトを設定することによりセットアップできます。ストレージクラブ分析で以下の処理が行われるように設定できます。

- バケットのコンテンツ全体を分析する。

  バケット内のすべてのオブジェクトの分析を受け取ります。
- プレフィックスとタグによってグループ化されたオブジェクトを分析する。

  プレフィックス、オブジェクトタグ、プレフィックスとタグの組み合わせによってオブジェクトを分析のためにグループ化するフィルターを設定できます。設定したフィルターごとに別個の分析を受け取ります。バケットあたり複数のフィルター (最大 1,000 個) を設定できます。
- 分析データをエクスポートする。

  バケットまたはフィルターのストレージクラス分析を設定するとき、分析データを毎日ファイルにエクスポートすることを選択できます。その日の分析がファイルに追加され、構成されたフィルターの履歴分析ログが形成されます。ファイルは、選択したエクスポート先で毎日更新されます。データのエクスポートを選択した場合、ファイルが書き込まれるエクスポート先バケットとオプションのエクスポート先プレフィックスを指定します。

Amazon S3 コンソール、REST API、AWS CLI、または AWS SDK を使用して、ストレージクラス分析を設定できます。

- Amazon S3 コンソールでストレージクラス分析を設定する方法については、「ストレージクラス分析の設定」を参照してください。
- Amazon S3 API を使用するには、AWS CLI または AWS SDK から PutBucketAnalyticsConfiguration REST API または同等の API を使用します。

ストレージクラス分析を使用する方法

データアクセスパターンを一定期間監視し、STANDARD_IA ストレージのライフサイクル管理の改善に役立つ情報を収集するには、ストレージクラス分析を使用します。フィルターを設定すると、フィルターに基づくデータ分析が 24 ~ 48 時間 Amazon S3 コンソールに表示され始め続けます。ただし、ストレージクラス分析はフィルタリングされたデータセットのアクセスパターンを 30 日以上監視し、分析のために情報を収集してから結果を生成します。最初の結果の後も分析は実行され続け、アクセスパターンが変わると結果が更新されます。

最初にフィルターを設定するとき、Simple Storage Service (Amazon S3) コンソールではデータの分析に時間がかかることがあります。

ストレージクラス分析は、フィルタリングされたオブジェクトデータセットのアクセスパターンを 30 日以上監視し、分析のために十分な情報を収集します。ストレージクラス分析が十分な情報を収集した後、Amazon S3 コンソールに分析の完了を示すメッセージが表示されます。

アクセス頻度の低いオブジェクトの分析を実行すると、ストレージクラス分析は、Amazon S3 にアプロードされてからの経過時間に基づいてグループ化された、フィルタリングされたオブジェクトセットを監視します。ストレージクラス分析は、フィルタリングされたデータセットの以下の要素を監視することで、経過時間グループのアクセス頻度が低いかどうかを判断します。

- STANDARD ストレージクラス内の 128 KB を超えるオブジェクト。
- 経過時間グループごとの合計ストレージ量の平均。
- 経過時間グループごとの平均転送 (アウト) バイト数 (頻度ではありません)。
- 分析のエクスポートデータには、ストレージクラスの分析に関連するデータのリクエストのみが含まれます。これにより、リクエストの数、および合計アップロードバイト数およびリクエストバイト数と、ストレージメトリクスで表示される数またはお客様独自の内部システムによって追跡される値との比較で違いが生じる可能性があります。
- 失敗した GET および PUT リクエストは分析でカウントされません。ただし、ストレージメトリクスには失敗したリクエストが含まれます。

取得したストレージの量。

Amazon S3 コンソールでは、フィルタリングされたデータセット内で観察期間内に取得されたストレージの量がグラフ化されます。

取得したストレージの割合。

Amazon S3 コンソールでは、フィルタリングされたデータセット内で観察期間内に取得されたストレージの割合もグラフ化されます。

このトピックですでに述べたように、アクセス頻度の低いオブジェクトの分析を実行すると、ストレージクラス分析は、Amazon S3 にアップロードされてからの経過時間に基づいてグループ化された、フィルタリングされたオブジェクトセットを監視します。ストレージクラス分析は、以下の定義済みオブジェクト経過時間グループを使用します。

- 15 日未満経過の Amazon S3 オブジェクト
- 15 ~ 29 日経過の Amazon S3 オブジェクト
- 30 ~ 44 日経過の Amazon S3 オブジェクト
- 45 ~ 59 日経過の Amazon S3 オブジェクト
- 60 ~ 74 日経過の Amazon S3 オブジェクト
- 75 ~ 89 日経過の Amazon S3 オブジェクト
- 90 ~ 119 日経過の Amazon S3 オブジェクト
- 120 ~ 149 日経過の Amazon S3 オブジェクト
- 150 ~ 179 日経過の Amazon S3 オブジェクト
- 180 ~ 364 日経過の Amazon S3 オブジェクト
- 365 ~ 729 日経過の Amazon S3 オブジェクト
- 730 日以上経過の Amazon S3 オブジェクト

通常は、分析結果の生成に十分な情報を収集するには、アクセスパターンの監視に約 30 日かかります。データ独自のアクセスパターンによっては、30 日より長くかかることもあります。ただし、フィルターを設定すると、フィルターに基づくデータ分析が 24 ~ 48 時間 Amazon S3 コンソールに表示され始め続けます。Amazon S3 コンソールでは、オブジェクト経過時間グループごとに分けられたオブジェクトアクセスの分析を毎日確認できます。

アクセス頻度の低いストレージの量。

Amazon S3 コンソールには、事前定義されたオブジェクトの経過時間グループごとにアクセスパターンがグループ分けされて表示されます。表示される [アクセスが頻繁] または [アクセスが頻繁で

はない] というテキストは、ライフサイクルの作成プロセスに役立つ視覚補助として提供されています。

ストレージクラス分析データをエクスポートする方法

ストレージクラス分析が分析レポートをカンマ区切り値 (CSV) フラットファイルにエクスポートすることを選択できます。レポートは毎日更新され、設定したオブジェクト経過時間グループフィルターに基づいています。Amazon S3 コンソールを使用すると、フィルターの作成時にレポートのエクスポートオプションを選択できます。データのエクスポートを選択した場合、ファイルが書き込まれるエクスポート先バケットとオプションのエクスポート先プレフィックスを指定します。データは、別のアカウントのエクスポート先バケットにエクスポートできます。エクスポート先バケットは、分析対象として設定したバケットと同じリージョンにある必要があります。

AWS アカウント がバケットを所有していることを確認し、定義された場所にあるバケットにオブジェクトを書き込むには、Amazon S3 への許可を付与するバケットポリシーをエクスポート先バケットで作成する必要があります。ポリシーの例については、「S3 インベントリおよび S3 分析に対するアクセス許可の付与」を参照してください。

ストレージクラス分析レポートを設定すると、24 時間後からエクスポートされたレポートが毎日生成され始めます。その後、Amazon S3 はモニタリングを続けて、エクスポートを毎日行います。

CSV ファイルは、スプレッドシートアプリケーションで開くか、Amazon QuickSight など、他のアプリケーションにインポートできます。Amazon QuickSight での Amazon S3 ファイルの使用については、Amazon QuickSight ユーザーガイドの「Create a Data Set Using Amazon S3 Files」を参照してください。

エクスポートされたファイルのデータは、次の例に示すように、オブジェクトの経過時間グループ内の日付でソートされます。ストレージクラスが STANDARD の場合、行には `ObjectAgeForSIATransition` 列と `RecommendedObjectAgeForSIATransition` 列のデータも含まれます。

Date	ConfigId	Filter	StorageClass	ObjectAge	ObjectCount	DataUploaded_MB	Storage_MB	DataRetrieved_MB	GetRequestCount	CumulativeAccessRatio	ObjectAgeForSIATransition	RecommendedObjectAgeForSIATransition
8/17/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/2/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/22/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
8/25/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/6/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/30/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/28/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/21/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/5/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		

レポートの最後に、オブジェクトの経過時間グループが ALL として示されます。ALL 行には、その日のすべての経過時間グループの累計 (128 KB 未満のオブジェクトを含む) が表示されます。

8/24/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0	000-014
9/3/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0.02426125	015-029
8/28/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0.03545875	015-029
8/17/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0	000-014
8/25/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0	000-014
9/6/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0.0209529	015-029
9/4/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0.02304819	015-029
8/22/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0	000-014
8/21/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0	000-014
8/30/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0.03073092	015-029
8/20/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0	000-014

次のセクションでは、レポートで使用される列について説明します。

エクスポートされたファイルのレイアウト

次の表では、エクスポートされたファイルのレイアウトについて説明します。

ストレージクラス分析の設定

Amazon S3 分析のストレージクラス分析ツールを使用することにより、ストレージアクセスパターンを分析し、適したデータをいつ適切なストレージクラスに移行すべきかを判断できます。ストレージクラス分析では、アクセス頻度の低い STANDARD ストレージをいつ STANDARD_IA (IA: 小頻度アクセス) ストレージクラスに移行すべきかを判断できるように、データアクセスパターンを確認します。STANDARD_IA の詳細については、Amazon S3 のよくある質問と「Amazon S3 ストレージクラスを使用する」を参照してください。

ストレージクラス分析は、分析するオブジェクトを設定することによりセットアップできます。ストレージクラブ分析で以下の処理が行われるように設定できます。

- バケットのコンテンツ全体を分析する。

  バケット内のすべてのオブジェクトの分析を受け取ります。
- プレフィックスとタグによってグループ化されたオブジェクトを分析する。

  プレフィックス、オブジェクトタグ、プレフィックスとタグの組み合わせによってオブジェクトを分析のためにグループ化するフィルターを設定できます。設定したフィルターごとに別個の分析を受け取ります。バケットあたり複数のフィルター (最大 1,000 個) を設定できます。
- 分析データをエクスポートする。

  バケットまたはフィルターのストレージクラス分析を設定するとき、分析データを毎日ファイルにエクスポートすることを選択できます。その日の分析がファイルに追加され、構成されたフィルターの履歴分析ログが形成されます。ファイルは、選択したエクスポート先で毎日更新されます。データのエクスポートを選択した場合、ファイルが書き込まれるエクスポート先バケットとオプションのエクスポート先プレフィックスを指定します。

Amazon S3 コンソール、REST API、AWS CLI、または AWS SDK を使用して、ストレージクラス分析を設定できます。

> **Important**
>
> ストレージクラス分析では、ONEZONE_IA または S3 Glacier Flexible Retrieval ストレージクラスへの移行に関するレコメンデーションを提供しません。
> 検出結果を .csv ファイルとしてエクスポートするようにストレージクラス分析を設定し、送信先バケットで AWS KMS key でデフォルトのバケット暗号化を使用する場合は、AWS KMS キーポリシーを更新して Amazon S3 に .csv ファイルを暗号化するための許可を付与する必要があります。手順については、「カスタマーマネージドキーを暗号化に使用するためのアクセス許可を Amazon S3 に付与する」を参照してください。

分析の詳細については、「Amazon S3 分析 – ストレージクラス分析」を参照してください。

S3 コンソールの使用

ストレージクラス分析を設定するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets (バケット)] リストで、ストレージクラス分析を設定するバケットの名前を選択します。
3. [Metrics] タブを選択します。
4. [Storage Class Analysis (ストレージクラス分析)] で、[Create analytics configuration (分析構成の作成)] を選択します。
5. フィルターの名前を入力します。バケット全体を分析する場合は、[Prefix (プレフィックス)] フィールドを空のままにします。
6. [Prefix (プレフィックス)] フィールドに、分析するオブジェクトの接頭辞のテキストを入力します。
7. タグを追加するには、[タグの追加] を選択します。タグのキーと値を入力します。1 つのプレフィックスと複数のタグを入力できます。
8. 必要に応じて、[CSV のエクスポート] で [有効化] を選択して、分析レポートをカンマ区切り値 (.csv) フラットファイルにエクスポートできます。ファイルを保存できる保存先のバケットを選択します。保存先バケットのプレフィックスを入力できます。ターゲットバケットは、分析を

セットアップするバケットと同じ AWS リージョン にある必要があります。ターゲットバケットは、別の AWS アカウント にある場合があります。

.csv ファイルの送信先バケットで KMS キーでデフォルトのバケット暗号化を使用する場合は、AWS KMS キーポリシーを更新して、Amazon S3 に .csv ファイルを暗号化するための許可を付与する必要があります。手順については、「カスタマーマネージドキーを暗号化に使用するためのアクセス許可を Amazon S3 に付与する」を参照してください。

9. [Create Configuration (設定の作成)] を選択します。

Amazon S3 は、Amazon S3 に書き込みアクセス許可を付与するバケットポリシーをターゲットバケットで作成します。これにより、バケットにエクスポート データを書き込むことができます。

バケットポリシーを作成しようとしたときにエラーが発生した場合、解決するための手順が表示されます。例えば、別の AWS アカウント 内のターゲットバケットを選択したが、バケットポリシーの読み書きを行う許可がない場合、次のメッセージが表示されます。ターゲットバケットの所有者に表示されたバケットポリシーをターゲットバケットに追加してもらう必要があります。ポリシーがターゲットバケットに追加されない場合、Amazon S3 にはターゲットバケットに書き込むアクセス許可がないため、エクスポートデータを取得できません。ソースバケットが現在のユーザー以外のアカウントによって所有されている場合、ポリシーでソースバケットの正しいアカウント ID に置き換える必要があります。

エクスポートされたデータとフィルターの動作の仕組みについては、「Amazon S3 分析 – ストレージクラス分析」を参照してください。

REST API の使用

REST API を使用してストレージクラス分析を設定するには、 PutBucketAnalyticsConfiguration を使用します。AWS CLI または AWS SDK で同等のオペレーションを使用することもできます。

次の REST API を使用してストレージクラス分析を操作できます。

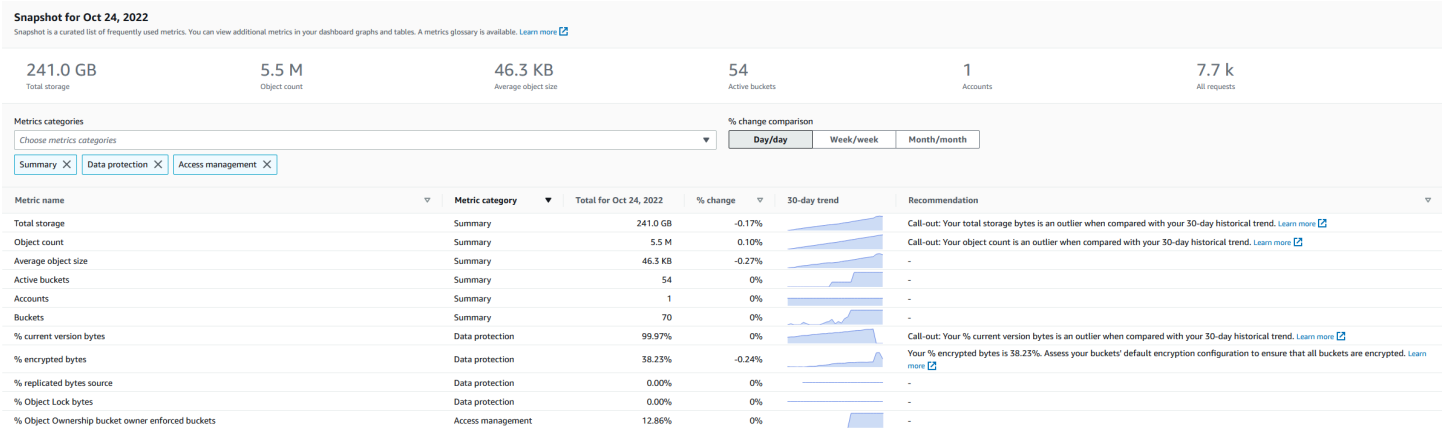
- DELETE バケット分析設定
- GET バケット分析設定
- List バケット分析設定

Amazon S3 ストレージレンズを使用してストレージのアクティビティと使用状況を評価する

Amazon S3 Storage Lens は、オブジェクトストレージおよびアクティビティを組織全体で可視化するために使用できるクラウドストレージ分析機能です。また、S3 Storage Lens は、メトリクスを分析して、ストレージコストを最適化し、データ保護に関するベストプラクティスを適用するために使用できるコンテキストに応じた推奨事項を提供します。

S3 Storage Lens メトリクスを使用して、概要の分析情報を生成できます。たとえば、組織全体でどれだけのストレージがあるか、または最も急速に成長しているバケットとプレフィックスは何かなどを把握することができます。S3 Storage Lens メトリクスを使用して、コスト最適化の機会を特定し、データ保護とアクセス管理のベストプラクティスを実装し、アプリケーションワークロードのパフォーマンスを向上させることができます。例えば、S3 ライフサイクルルールがないバケットを特定して、7 日以上経過した不完全なマルチパートアップロードを中止できます。また、S3 レプリケーションや S3 バージョニングの使用など、データ保護のベストプラクティスに従っていないバケットを特定することもできます。

S3 ストレージレンズはメトリクスを集約し、Amazon S3 コンソールの [Buckets] (バケット) ページの [Account snapshot] (アカウントスナップショット) セクションにこの情報を表示します。S3 Storage Lens は、インサイトと傾向を可視化したり、外れ値にフラグ付けしたり、ストレージコストの最適化やデータ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取るために使用できるインタラクティブダッシュボードも提供します。ダッシュボードには、組織、アカウント、AWS リージョン、バケット、オブジェクト、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成して可視化できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。



S3 ストレージレンズのメトリクスと機能

S3 ストレージレンズでは、毎日更新されるインタラクティブなデフォルトダッシュボードが利用できます。S3 ストレージレンズではこのダッシュボードが事前定義されており、アカウント全体の要約されたインサイトとトレンドを視覚化し、S3 コンソールで毎日更新します。このダッシュボードのメトリクスは、[Buckets] (バケット) ページのアカウントスナップショットにも要約されます。詳細については、「デフォルトのダッシュボード」を参照してください。

他のダッシュボードを作成し、AWS リージョン、S3 バケットまたはアカウント (AWS Organizations 用) でスコープを設定するには、S3 ストレージレンズダッシュボード設定を作成します。AmazonS3 コンソール、AWS Command Line Interface、(AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して S3 Storage Lens ダッシュボード設定を作成および管理できます。S3 Storage Lens ダッシュボードを作成または編集するとき、ダッシュボードのスコープとメトリクスの選択を定義します。

S3 ストレージレンズでは、無料のメトリクスと、追加料金でアップグレードできるアドバンストメトリクスとレコメンデーションをご利用になれます。アドバンストメトリクスとレコメンデーションをご利用になれば、ストレージに関するインサイトが得られる追加のメトリクスや機能にアクセスできます。これらの機能には、高度なメトリクスカテゴリ、プレフィックス集約、コンテキストに応じた推奨事項、Amazon CloudWatch パブリッシングなどがあります。プレフィックス集約とコンテキストに応じた推奨事項は、Amazon S3 コンソールでのみ使用できます。S3 Storage Lens の料金の詳細については、Amazon S3 の料金を参照してください。

メトリクスのカテゴリ

無料層とアドバンスト層で、メトリクスはコストの最適化やデータ保護などの主要なユースケースに合わせてカテゴリに分類されます。無料のメトリクスには、概要、コスト最適化、データ保護、アクセス管理、パフォーマンス、イベントメトリクスが含まれます。アドバンストメトリクスとレコメンデーションにアップグレードすると、高度なコスト最適化とデータ保護メトリクスを有効にできます。これらの高度なメトリクスを使用すると、S3 ストレージのコストをさらに削減し、データ保護スタンスを向上させることができます。アクティビティメトリクスと詳細なステータスコードメトリクスを有効にして、S3 バケットにアクセスしているアプリケーションワークロードのパフォーマンスを向上させることもできます。無料のメトリクスと高度なメトリクスのカテゴリの詳細については、「メトリクスの選択」を参照してください。

暗号化、S3 オブジェクトロック、または S3 バージョニングが有効になっているバケットの割合を分析するなど、S3 のベストプラクティスに基づいてストレージを評価できます。また、潜在的なコスト削減の機会を特定することもできます。例えば、S3 ライフサイクルルール数メトリクスを使用

して、ライフサイクルの有効期限または移行ルールがないバケットを特定できます。また、バケットごとのリクエストアクティビティを分析して、コストのより低いストレージクラスにオブジェクトの移行が可能なバケットを見つけることもできます。詳細については、「Amazon S3 ストレージレンズメトリクスのユースケース」を参照してください。

メトリクスのエクスポート

S3 コンソールのダッシュボードで表示するだけでなく、メトリクスを CSV または Parquet 形式で S3 バケットにエクスポートして、選択した分析ツールでさらに分析することができます。詳細については、「データエクスポートで Amazon S3 Storage Lens のメトリクスを確認する」を参照してください。

Amazon CloudWatch パブリッシング

S3 ストレージレンズの使用状況とアクティビティメトリクスを Amazon CloudWatch に公開し、CloudWatch ダッシュボードで運用状況を一元的に表示することができます。また、アラームやトリガーアクション、メトリクス計算、異常検出などの CloudWatch 機能を使用して、S3 ストレージレンズメトリクスをモニタリングして対処することができます。さらに CloudWatch API オペレーションにより、サードパーティープロバイダーを含むアプリケーションが S3 ストレージレンズメトリクスにアクセスできるようになります。CloudWatch の公開オプションは、S3 ストレージレンズのアドバンストメトリクスとレコメンデーションにアップグレードされたダッシュボードで利用可能です。CloudWatch での S3 ストレージレンズメトリクスの詳細については、「CloudWatch で S3 Storage Lens のメトリクスをモニタリング」を参照してください。

S3 ストレージレンズの使用の詳細については、次のトピックを参照してください。

トピック

- Amazon S3 ストレージレンズを理解する
- AWS Organizations での Amazon S3 ストレージレンズの使用
- Amazon S3 ストレージレンズアクセス許可
- Amazon S3 ストレージレンズでのメトリクスの表示
- Amazon S3 ストレージレンズメトリクスのユースケース
- Amazon S3 Storage Lens のメトリクスに関する用語集
- コンソールと API を使用した Amazon S3 ストレージレンズの使用
- S3 Storage Lens グループの使用

Amazon S3 ストレージレンズを理解する

> **Important**
>
> Amazon S3 では、Amazon S3 のすべてのバケットに対する基本レベルの暗号化として、Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) が適用されるようになりました。2023 年 1 月 5 日以降、Amazon S3 にアップロードされるすべての新しいオブジェクトは、追加費用なしで、パフォーマンスに影響を与えずに自動的に暗号化されます。S3 バケットのデフォルト暗号化設定と新しいオブジェクトのアップロードのための自動暗号化ステータスは、AWS CloudTrail ログ、S3 インベントリ、S3 ストレージレンズ、Amazon S3 コンソール、および AWS Command Line Interface と AWS SDK の追加の Amazon S3 API レスポンスヘッダーとして利用できるようになりました。詳細については、「デフォルトの暗号化のよくある質問」を参照してください。

Amazon S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージ分析機能です。S3 ストレージレンズメトリクスを使用することで、組織全体でどれだけのストレージがあるか、または最も急速に成長しているバケットとプレフィックスは何かなどの、要約されたインサイトを生成できます。S3 ストレージレンズメトリクスを使用して、コスト最適化の機会を特定し、データ保護とセキュリティのベストプラクティスを実装し、アプリケーションワークロードのパフォーマンスを向上させることもできます。例えば、S3 ライフサイクルルールがないバケットを特定して、7 日以上経過した未完了のマルチパートアップロードを有効期限切れにできます。また、S3 レプリケーションや S3 バージョニング使用など、データ保護のベストプラクティスに従っていないバケットを特定することもできます。また、S3 ストレージレンズは、メトリクスを分析して、ストレージコストを最適化し、データ保護に関するベストプラクティスを適用するために使用できるコンテキストに応じた推奨事項を提供します。

S3 ストレージレンズはメトリクスを集約し、Amazon S3 コンソールの [Buckets] (バケット) ページの [Account snapshot] (アカウントスナップショット) セクションにこの情報を表示します。S3 Storage Lens は、インサイトと傾向を可視化したり、外れ値にフラグ付けしたり、ストレージコストの最適化やデータ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取るために使用できるインタラクティブダッシュボードも提供します。ダッシュボードには、組織、アカウント、AWS リージョン、バケット、オブジェクト、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成して可視化できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。AmazonS3 コンソール、AWS Command Line Interface、(AWS CLI)、AWS

SDK、または Amazon S3 REST API を使用して S3 Storage Lens ダッシュボードを作成および管理できます。

S3 ストレージレンズの概念と用語

このセクションは、Amazon S3 ストレージレンズを適切に理解し使用するために不可欠な、用語と概念に関する解説で構成されています。

トピック

ダッシュボード設定

S3 ストレージレンズでは、ユーザーに代わってメトリクスを集計するのに必要なプロパティを含むダッシュボード設定が必要となります。これらは、単一のダッシュボード、もしくはエクスポートのために使用されます。設定を作成するときは、ダッシュボード名とホームリージョンを選択します。これらはダッシュボードの作成後に変更できません。オプションでタグを追加し、CSV または Parquet 形式でメトリクスをエクスポートするように設定できます。

ダッシュボード設定では、ダッシュボードのスコープとメトリクスの選択も定義します。スコープには、組織アカウントのすべてのストレージ、またはリージョン、バケット、アカウントでフィルタリングされたセクションのストレージを含めることができます。メトリクスの選択を設定する際、無料のメトリクスまたは追加料金でアップグレードできるアドバンストメトリクスとレコメンデーションのいずれかを選択します。アドバンストメトリクスとレコメンデーションにより、その他のメトリクスや機能にアクセスできます。これらの機能には、高度なメトリクスカテゴリ、プレフィックスレ

ベルでの集約、コンテキストに応じた推奨事項、Amazon CloudWatch パブリッシングなどがあります。S3 ストレージレンズの料金の詳細については、Amazon S3 の料金を参照してください。

デフォルトのダッシュボード

コンソールの S3 ストレージレンズのデフォルトダッシュボードの名前は、default-account-dashboard です。S3 ではこのダッシュボードが事前定義されており、アカウント全体の要約されたインサイトとトレンドを視覚化し、S3 コンソールで毎日更新します。デフォルトのダッシュボードのスコープ設定を変更することはできませんが、選択するメトリクスを、無料メトリクスからアドバンストメトリクスとレコメンデーションにアップグレードすることはできます。オプションで、メトリクスのエクスポートを設定したり、ダッシュボードを無効にしたりもできます。ただし、デフォルトのダッシュボードを削除することはできません。

> **Note**
>
> デフォルトのダッシュボードを無効にすると、更新されなくなります。S3 ストレージレンズダッシュボード、メトリクスエクスポート、または S3 [バケット] ページのアカウントスナップショットで、新しい日次メトリクスを受け取ることはなくなります。ダッシュボードで高度なメトリクスや推奨事項を使用していても、請求は発生しなくなります。データクエリの 14 日の有効期限が切れるまでは、ダッシュボードで履歴データを引き続き表示できます。アドバンストメトリクスとレコメンデーションを有効にしている場合、この期間は 15 か月です。有効期限内にダッシュボードを再度有効にすれば、履歴データにアクセスできます。

ダッシュボード

追加の S3 ストレージレンズダッシュボードを作成し、AWS リージョン、S3 バケット、またはアカウント (AWS Organizations の場合) によってスコープを設定できます。S3 ストレージレンズダッシュボードを作成または編集するとき、ダッシュボードのスコープとメトリクスの選択を定義します。S3 ストレージレンズでは、無料のメトリクスと、追加料金でアップグレードできるアドバンストメトリクスとレコメンデーションをご利用になれます。アドバンストメトリクスとレコメンデーションをご利用になれば、ストレージに関するインサイトが得られる追加のメトリクスや機能にアクセスできます。これらには、高度なメトリクスカテゴリ、プレフィックスレベルでの集約、コンテキストに応じた推奨事項、Amazon CloudWatch パブリッシングなどがあります。S3 Storage Lens の料金の詳細については、「Amazon S3 の料金」を参照してください。

また、ダッシュボードを無効化または削除することも可能です。ダッシュボードを無効にすると、そのダッシュボードは更新されなくなるので、日次で新しいメトリクスが生成されることもなくなり

ます。14 日の有効期限の間は、履歴データを表示できます。そのダッシュボードでアドバンストメトリクスとレコメンデーションを有効にした場合、この期間は 15 か月です。有効期限内にダッシュボードを再度有効にすれば、履歴データにアクセスできます。

ダッシュボードを削除すると、それまでの設定内容はすべて失われます。削除されたダッシュボードからは、それ以後の新しい日次のメトリクスは受信できなくなり、そのダッシュボードに関連付けられた履歴データにもアクセスできなくなります。削除したダッシュボードの履歴データにアクセスする場合は、同じホームリージョンで同じ名前を使用しながら、新たなダッシュボードを作成する必要があります。

> **Note**
>
> - S3 ストレージレンズをご使用のお客様は、ホームリージョンごとに最大 50 個のダッシュボードを作成いただけます。
> - 組織レベルのダッシュボードは、リージョンのスコープにのみ制限できます。

アカウントスナップショット

S3 ストレージレンズの [Account snapshot] (アカウントスナップショット) は、デフォルトのダッシュボードからメトリクスを要約し、S3 コンソールの ([Buckets] (バケット) ページに、ストレージ合計、オブジェクト数、および平均オブジェクトサイズを表示します。このアカウントスナップショットでは、[Buckets] (バケット) ページから移動することなく、ストレージに関するインサイトにすばやくアクセスできます。アカウントスナップショットから、インタラクティブな S3 ストレージレンズダッシュボードにワンクリックでアクセスすることもできます。

ダッシュボードでは、インサイトと傾向を可視化したり、外れ値にフラグ付けしたりできます。また、ストレージコストの最適化や、データ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取ることもできます。ダッシュボードには、組織、アカウント、バケット、オブジェクト、またはプレフィックスレベルでインサイトを生成できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。

default-account-dashboard のスコープは、アカウントスナップショットにリンクされているため、変更できません。ただし、default-account-dashboard のメトリクス選択を、無料のメトリクスから有料のアドバンストメトリクスとレコメンデーションにアップグレードできます。アップグレード後、すべてのリクエスト、アップロードされたバイト数、およびダウンロードされたバイト数を S3 ストレージレンズのアカウントスナップショットで表示できます。

> **Note**
>
> デフォルトのダッシュボードを無効にすると、アカウントスナップショットは更新されなくなります。アカウントスナップショットでのメトリクスの表示を再開するには、default-account-dashboard を再度有効にします。

メトリクスのエクスポート

S3 ストレージレンズでは、1 つのファイルとしてメトリクスのエクスポートを作成します。このファイルには、S3 ストレージレンズの設定で指定された、すべてのメトリクスが含まれています。この情報は、CSV または Parquet 形式で毎日生成され、S3 バケットに送信されます。メトリクスをエクスポートすれば、お好みのメトリクスツールでさらに分析することができます。メトリクスのエクスポート用の S3 バケットは、S3 ストレージレンズ設定と同じリージョンに存在する必要があります。ダッシュボードの設定を編集することで、S3 コンソールから S3 ストレージレンズメトリクスのエクスポートを生成できます。AWS CLI および AWS SDK を使用してメトリクスのエクスポートを設定することもできます。

ホームリージョン

ホームリージョンとは、特定のダッシュボード設定のすべての S3 ストレージレンズメトリクスが保存される AWS リージョン です。S3 ストレージレンズのダッシュボード設定を作成する際には、ホームリージョンを選択する必要があります。いったん選択したホームリージョンは変更できません。また、Storage Lens グループを作成する場合は、Storage Lens ダッシュボードと同じホームリージョンを選択することをお勧めします。

> **Note**
>
> 次のいずれかの地域をホームリージョンとして選択できます。
>
> - 米国東部 (バージニア北部) – `us-east-1`
> - 米国東部 (オハイオ) – `us-east-2`
> - 米国西部 (北カリフォルニア) – `us-west-1`
> - 米国西部 (オレゴン) – `us-west-2`
> - アジアパシフィック (ムンバイ) – `ap-south-1`
> - アジアパシフィック (ソウル) – `ap-northeast-2`
> - アジアパシフィック (シンガポール) – `ap-southeast-1`

- アジアパシフィック (シドニー) – ap-southeast-2
- アジアパシフィック (東京) – ap-northeast-1
- カナダ (中部) – ca-central-1
- 中国 (北京) – cn-north-1
- 中国 (寧夏) – cn-northwest-1
- 欧州 (フランクフルト) – eu-central-1
- 欧州 (アイルランド) – eu-west-1
- 欧州 (ロンドン) – eu-west-2
- 欧州 (パリ) – eu-west-3
- 欧州 (ストックホルム) – eu-north-1
- 南米 (サンパウロ) – sa-east-1

保持期間

S3 Storage Lens のメトリクスは保持されるため、過去の傾向を確認したり、ストレージの使用状況やアクティビティの変動を時間の経過を追って比較するために使用できます。Amazon S3 ストレージレンズのメトリクスをクエリに使用することで、過去の傾向を確認し、ストレージの使用量とアクティビティの違いを時系列で比較することができます。

すべての S3 ストレージレンズメトリクスは 15 か月間保持されます。ただし、メトリクスは特定の期間のクエリでのみ使用できます。これは、選択したメトリクスの種類によって異なります。この期間は変更できません。無料のメトリクスは 14 日間、アドバンストメトリクスは 15 か月間のクエリで利用可能です。

メトリクスのカテゴリ

無料層とアドバンスト層で、S3 ストレージレンズメトリクスはコストの最適化やデータ保護などの主要なユースケースに合わせてカテゴリに分類されます。無料のメトリクスには、概要、コスト最適化、データ保護、アクセス管理、パフォーマンス、イベントメトリクスが含まれます。高度なメトリクスとレコメンデーションにアップグレードすると、追加のコスト最適化とデータ保護メトリクスを有効にして、S3 ストレージのコストをさらに削減し、データ保護を確実にできます。また、アクティビティメトリクスと詳細なステータスコードメトリクスを有効にして、アプリケーションワークロードのパフォーマンスを向上させために使用することもできます。

次のリストは、すべての無料および高度なメトリクスのカテゴリを示しています。各カテゴリに含まれる個々のメトリクスの完全なリストについては、「メトリクスの用語集」を参照してください。

サマリーメトリクス

サマリーメトリクスは、ストレージの合計バイト数やオブジェクト数など、S3 ストレージに関する一般的な情報を提供します。

コスト最適化メトリクス

コスト最適化メトリクスは、ストレージコストの管理と最適化に役立つインサイトを提供します。例えば、7 日以上経過した不完全なマルチパートアップロードを特定できます。

アドバンストメトリクスとレコメンデーションにより、高度なコスト最適化メトリクスを有効にできます。これらのメトリクスには、バケットごとの有効期限と移行 S3 ライフサイクルルール数を取得するために使用できる S3 ライフサイクルルールカウントメトリクスが含まれています。

データ保護メトリクス

データ保護メトリクスは、暗号化や S3 バージョニングなどのデータ保護機能に関するインサイトを提供します。これらのメトリクスを使用して、データ保護のベストプラクティスに従っていないバケットを特定できます。例えば、デフォルトの暗号化に AWS Key Management Service キー (SSE-KMS) を使用していないバケットや、S3 バージョニングを使用していないバケットを特定できます。

アドバンストメトリクスとレコメンデーションにより、高度なデータ保護メトリクスを有効にできます。これらのメトリクスには、バケットごとのレプリケーションルール数メトリクスが含まれます。

アクセス管理メトリクス

アクセス管理メトリクスは、S3 オブジェクト所有権に関するインサイトを提供します。これらのメトリクスを使用して、バケットが使用しているオブジェクト所有権設定を確認できます。

イベントメトリクス

イベントメトリクスは、S3 イベント通知に関するインサイトを提供します。イベントメトリクスでは、どのバケットに S3 イベント通知が設定されているかを確認できます。

パフォーマンスメトリクス

パフォーマンスメトリクスは、S3 Transfer Acceleration のインサイトを提供します。パフォーマンスメトリクスでは、どのバケットで Transfer Acceleration が有効になっているかを確認できます。

アクティビティメトリクス (アドバイスト)

ダッシュボードを高度なメトリクスとレコメンデーションにアップグレードすると、アクティビティメトリクスを有効にできます。アクティビティメトリクスは、ストレージがどのようにリクエストされたか (例えば、ALL リクエスト、GET リクエスト、PUT リクエストなど)、アップロードまたはダウンロードされたバイト数、エラーなどに関する詳細を提供します。

プレフィックスレベルのアクティビティメトリクスを使用すると、使用頻度の低いプレフィックスを特定できるため、S3 ライフサイクルを使用してより最適なストレージクラスに移行できます。

詳細なステータスコードメトリクス (アドバンスト)

ダッシュボードを高度なメトリクスとレコメンデーションにアップグレードすると、詳細なステータスコードメトリクスを有効にできます。詳細なステータスコードメトリクスにより、403 Forbidden や 503 Service Unavailable などの HTTP ステータスコードに関するインサイトが得られ、アクセスやパフォーマンスの問題のトラブルシューティングに使用できます。例えば、403 Forbidden エラー数メトリクスを見ると、適切なアクセス権限が適用されていない状態でバケットにアクセスしているワークロードを特定できます。

プレフィックスレベルの詳細なステータスコードメトリクスを使用すると、プレフィックスごとの HTTP ステータスコードの発生に関する理解を深めることができます。たとえば、503 エラーカウントメトリクスを使用すると、データ取り込み中にスロットリングリクエストを受信したプレフィックスを特定できます。

レコメンデーション

S3 ストレージレンズでは、ストレージの最適化に役立つ自動化された推奨事項が提供されます。それぞれの推奨事項は、関連のあるメトリクスとともに、文脈に応じた形で S3 ストレージレンズダッシュボード上に表示されます。履歴データは、推奨事項の対象になりません。推奨事項は、最近に発生した事象に関連しています。推奨事項は、関連性が存在する場合にのみ表示されます。

S3 ストレージレンズの推奨事項は、次のような形式で構成されます。

- 提案

  提案は、ストレージとアクティビティでのトレンドに関する注意を促します。この場合、ストレージコストの最適化や、データ保護のベストプラクティスを実施できる可能性があります。Amazon S3 ユーザーガイドで、この提案に関するトピックを参照できます。同時に、S3 Storage Lens ダッシュボードから特定のリージョン、バケット、またはプレフィックスにドリルダウンすることができます。
- コールアウト

コールアウトは、ストレージおよびアクティビティに関し、一定期間に発生して関心を向けるべき異常を通知するための推奨事項です。これらの結果には、さらなる注意や監視が必要となる場合があります。

- 外れ値のコールアウト

  S3 ストレージレンズでは、最近の 30 日間の傾向に基づいて判断した、外れ値のメトリクスに関するコールアウトが利用できます。外れ値は、Z-score とも呼ばれる標準スコアを使用して算出されています。このスコアでは、当日のメトリクスが、同じメトリクスの最近 30 日間の平均から減算されます。次に、当日のメトリクスが、同じメトリクスの最近 30 日間の標準偏差により除算されます。結果として得られるスコアは、通常 -3 ~ +3 の間の値となります。この数値は、当該日のメトリクスの、平均値からの標準偏差を表しています。

  S3 ストレージレンズでは、スコアが 2 から -2 の範囲を上下に超えるメトリクスは外れ値と見なされます。理由は、この値が正規分布データの 95% よりも高い、あるいは低くなるためです。

- 顕著な変化についてのコールアウト

  顕著な変化についてのコールアウトは、変化する頻度が低いことが想定されているメトリクスに対し適用されます。このため、この値の計算では、外れ値の計算よりも高い感度が設定されます。この感度には通常、前日、前週、または前月に比べて +/-20% の範囲が指定されます。

  ストレージの使用状況とアクティビティに関するコールアウトへの対応 – 顕著な変化についてのコールアウトを受け取った場合でも、必ずしも問題が存在することにはなりません。コールアウトは、ストレージで予定された変更の結果である可能性もあります。たとえば、多数の新しいオブジェクトの追加や多数のオブジェクトの削除を最近行っていたり、それと同様の計画的な変更を行ったりした場合など、このコールアウトの原因となり得ます。

  ダッシュボード上に、顕著な変化についてのコールアウトが表示される場合は、それを記録した上で、その原因が、最近の状況によって説明できるかどうかを確認します。原因が見つからない場合は、S3 ストレージレンズダッシュボードを使用して詳細をドリルダウンしながら、変動要因となっている特定のリージョン、バケット、またはプレフィックスを調査します。

- リマインダー

  リマインダーにより、Amazon S3 の動作についてのインサイトが提供されます。リマインダーは、ストレージコストの削減や、データ保護のベストプラクティスの適用のための S3 の機能の使用方法について、詳しく知るために利用できます。

メトリクスの選択

S3 ストレージレンズのダッシュボードとエクスポートでは、メトリクスを、無料のメトリクスとアドバンストメトリクスとレコメンデーションという 2 種類から選択できます。

- 無料のメトリクス

  S3 ストレージレンズの無料のメトリクスは、すべてのダッシュボードと設定で使用できます。無料のメトリクスには、アカウント内のバケットやオブジェクトの数など、ストレージに関連するメトリクスが含まれています。無料のメトリクスには、ストレージが S3 のベストプラクティスに従って設定されているかどうかを調べるために使用できるユースケースベースのメトリクス (コスト最適化やデータ保護メトリクスなど) も含まれています。すべての無料メトリクスは毎日収集されます。データは、14 日間クエリで利用可能です。無料のメトリクスについては、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

- アドバンストメトリクスとレコメンデーション

  S3 ストレージレンズでは、すべてのダッシュボードと構成に対して無料のメトリクスを提供していますが、さらに、アドバンストメトリクスとレコメンデーションにアップグレードできるオプションも用意されています。追加の変更が適用されます。詳細については、「Amazon S3 の料金」を参照してください。

  アドバンストメトリクスとレコメンデーションには、無料のメトリクスに含まれるすべてのメトリクスに加えて、高度なデータ保護やコスト最適化のメトリクス、アクティビティメトリクス、詳細なステータスコードメトリクスなどの追加のメトリクスが含まれます。また、アドバンストメトリクスとレコメンデーションは、ストレージの最適化に役立つ推奨事項も提供します。それぞれの推奨事項は、関連するメトリクスとともに、文脈に応じた形でダッシュボード上に表示されます。

  また、高度なメトリクスとレコメンデーションには以下の機能も含まれています。

  - アドバンストメトリクス — 追加のメトリクスを生成します。アドバンストメトリクスカテゴリの完全なリストについては、「メトリクスのカテゴリ」を参照してください。メトリクスの一覧については、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。
  - Amazon CloudWatch パブリッシング – S3 Storage Lens メトリクスを CloudWatch に公開して、CloudWatch ダッシュボードで運用の健全性を一元的に把握できます。また、アラームやトリガーアクション、メトリクス計算、異常検出などの CloudWatch API オペレーションと機能を使用して、S3 ストレージレンズメトリクスをモニタリングして対処することができます。詳細については、「CloudWatch で S3 Storage Lens のメトリクスをモニタリング」を参照してください。

- プレフィックス集計 – プレフィックスレベルでメトリクスを集計します。プレフィックス集計を有効にすると、ダッシュボード設定に含まれるすべてのメトリクスがプレフィックスレベルで拡張されます。メトリクスは、設定したしきい値を満たすプレフィックスに対してのみ生成されます。プレフィックスレベルで適用できるメトリクスは、バケットレベルの設定とルールカウントメトリクスを除き、プレフィックス集計で利用できることに注意してください。プレフィックスレベルのメトリクスは CloudWatch に公開されません。
- Storage Lens グループ集計 — Storage Lens グループレベルでメトリクスを収集します。[高度なメトリクスとレコメンデーション] と [Storage Lens グループ集計] を有効にすると、どのStorage Lens グループを Storage Lens ダッシュボードに含めたり除外したりするかを指定できます。少なくとも 1 つの Storage Lens のグループを指定する必要があります。指定する Storage Lens グループは、ダッシュボードアカウントの指定されたホームリージョン内にも存在している必要があります。Storage Lens グループレベルのメトリクスは CloudWatch に公開されません。

すべてのアドバンストメトリクスは毎日収集されます。データは最長 15 か月間クエリで使用できます。S3 ストレージレンズによって集計されるストレージメトリクスの詳細については、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

> **Note**
>
> 推奨事項が利用できるのは、Amazon S3 コンソールで S3 Storage Lens ダッシュボードを使用している場合だけです。

S3 ストレージレンズと AWS Organizations

AWS Organizations は、1 つの組織階層に属する、すべての AWS アカウント を集約するための AWS のサービス です。Amazon S3 Storage Lens を AWS Organizations と連携させることで、Amazon S3 ストレージ全体でのオブジェクトによるストレージとそのアクティビティを、単一的なビューで表示することができます。

詳細については、「AWS Organizations での Amazon S3 ストレージレンズの使用」を参照してください。

- 信頼されたアクセス

  組織内のすべてのメンバーアカウントから、ストレージのメトリクスと使用状況に関するデータを集約するためには、組織の管理アカウントを使用して、S3 ストレージレンズの信頼されたアクセ

スを有効にする必要があります。その後、管理アカウントを直接使用するか、組織内の他のアカウントに管理者アクセス権限を委任すること (委任管理者) により、組織で使用するダッシュボードまたはエクスポートを作成できるようになります。

S3 ストレージレンズの信頼できるアクセスはいつでも無効にできます。これにより S3 ストレージレンズは、組織に関するメトリクスの集約を停止します。

- 委任管理者

  組織の S3 ストレージレンズのダッシュボードとメトリクスの作成は、AWS Organizations の管理アカウントを使用するか、組織内の他のアカウントに委任管理者権限を付与することで実行できます。委任管理者の登録はいつでも解除できます。また、委任管理者の登録を解除すると、委任管理者によって作成されたすべての組織レベルのダッシュボードで、ストレージのメトリクスの新たな集計が自動的に停止されます。

詳細については、AWS Organizations ユーザーガイドの 「Amazon S3 ストレージレンズと AWS Organizations」 を参照してください。

Amazon S3 ストレージレンズでのサービスにリンクされたロール

AWS Organizations での信頼されたアクセスに加えて、Amazon S3 ストレージレンズでは、AWS Identity and Access Management (IAM) の、サービスにリンクされたロールも使用できます。サービスにリンクされたロールは S3 ストレージレンズに直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、S3 Storage Lens によって事前定義されています。このロールには、組織内のメンバーアカウントからストレージとアクティビティに関する日次のメトリクスを収集するために必要な、すべてのアクセス権限が含まれています。

詳細については、「Amazon S3 ストレージレンズでのサービスにリンクされたロールの使用」を参照してください。

AWS Organizations での Amazon S3 ストレージレンズの使用

Amazon S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージの分析機能です。S3 ストレージレンズメトリクスを使用することで、組織全体でどれだけのストレージがあるか、または最も急速に成長しているバケットとプレフィックスは何かなどの、要約されたインサイトを生成できます。S3 ストレージレンズメトリクスを使用して、コスト最適化の機会を特定し、データ保護とセキュリティのベストプラクティスを実装し、アプリケーションワークロードのパフォーマンスを向上させることもできます。例えば、S3 ライフサイクルルールがないバケットを特定して、7 日以上経過した未完了のマルチパートアップロードを有効期限切れにできます。また、S3 レプリケーションや S3 バージョニン

グ使用など、データ保護のベストプラクティスに従っていないバケットを特定することもできます。また、S3 Storage Lens は、メトリクスを分析して、ストレージコストを最適化し、データ保護に関するベストプラクティスを適用するために使用できるコンテキストに応じた推奨事項を提供します。

Amazon S3 ストレージレンズを使用することで、AWS Organizations の階層に属しているすべての AWS アカウント から、ストレージのメトリクスと使用状況に関するデータを収集できます。これを行うには、AWS Organizations を使用している必要があります。また、AWS Organizations 管理アカウントを使用して、S3 ストレージレンズに対し、信頼されたアクセスを有効にする必要があります。

信頼されたアクセスを有効にすると、組織内のアカウントに対し、委任管理者の権限を追加できるようになります。これらのアカウントは、組織全体のストレージメトリクスとユーザーデータを収集する S3 ストレージレンズの設定とダッシュボードを作成できます。

信頼されたアクセスの有効化の詳細については、AWS Organizations ユーザーガイドの「Amazon S3 ストレージレンズと AWS Organizations」を参照してください。

トピック

S3 ストレージレンズのための信頼されたアクセスを有効にする

信頼されたアクセスが有効化された Amazon S3 ストレージレンズでは、AWS Organizations API オペレーションを通じて、AWS Organizations の階層、メンバーシップ、および構造にアクセスできるようになります。S3 ストレージレンズは、組織全体の構造で信頼されたサービスとして認識されます。

ダッシュボードの設定が作成されるたびに、組織の管理アカウントまたは委任管理者アカウントのために、サービスにリンクされたロールが S3 ストレージレンズにより作成されます。サービスにリンクされたロールにより、S3 Storage Lens が次の操作を実行できます。

- 組織を説明する
- アカウントを一覧表示する
- 組織の AWS のサービス アクセスリストを検証する
- 組織の委任管理者を取得する

S3 ストレージレンズは、組織のアカウントのクロスアカウントストレージ使用状況およびアクティビティメトリクスを収集するためのアクセスを確実に有しているようにできます。詳細については、「Amazon S3 ストレージレンズでのサービスにリンクされたロールの使用」を参照してください。

信頼されたアクセスを有効にしておくと、組織内のアカウントに対し、委任管理者権限を割り当てることができます。サービスに対する委任管理者としてマークされたアカウントには、組織内で、すべての読み取り専用 API オペレーションにアクセスするための承認が与えられます。このアクセス権により、委任管理者には組織のメンバーと構造に対する可視性が提供され、彼らも S3 ストレージレンズダッシュボードを作成できるようになります。

> **Note**
>
> Amazon S3 ストレージレンズに対し信頼されたアクセスを有効にできるのは、管理アカウントのみです。

S3 ストレージレンズのための信頼されたアクセスを無効にする

信頼されたアクセスを無効にすると、S3 ストレージレンズはアカウントレベルでのみ機能するように制限されます。さらに、各アカウント保有者は S3 ストレージレンズの機能に対して、組織全体ではなくアカウントの範囲内に限りアクセスできるようになります。信頼されたアクセスを必要とするダッシュボードは更新されなくなりますが、データがクエリで利用可能な期間中は履歴データは保持されます。

> **Note**
>
> - また、S3 ストレージレンズの信頼されたアクセスを無効にすると、すべての組織レベルのダッシュボードで、ストレージのメトリクスの収集と集計が自動的に停止されます。
> - 管理者、および委任管理者アカウントは、データがクエリ用に利用可能な期間中、終了した組織レベルのダッシュボードの履歴データを引き続き表示できます。

S3 ストレージレンズでの委任管理者の登録

組織の管理アカウントまたは委任管理者アカウントからは、組織レベルのダッシュボードを作成できます。委任管理者アカウントでは、管理アカウントを除く他のアカウントに対し、組織レベルのダッ

シュボードの作成を許可できます。他のアカウントを組織の委任管理者として登録および登録解除できるのは、その組織の管理者アカウントのみです。

委任管理者を、Amazon S3 コンソールを使用して登録するには、「S3 Storage Lens のための委任された管理者の登録」を参照してください。

また、委任管理者は、管理アカウントから AWS Organizations の REST API、AWS CLI、または SDK を使用することでも登録できます。詳細については、AWS Organizations API リファレンスの「RegisterDelegatedAdministrator」を参照してください。

> **Note**
>
> AWS Organizations の REST API、AWS CLI、または SDK を使用して委任管理者を指定するには、その前に、EnableAWSOrganizationsAccess オペレーションを呼び出す必要があります。

S3ストレージレンズでの委任管理者の登録を解除する

委任管理者アカウントは、登録を解除することもできます。委任管理者アカウントでは、管理アカウントを除く他のアカウントに対し、組織レベルのダッシュボードの作成を許可できます。組織の管理アカウントのみが、その組織内の委任管理者のアカウントの登録を解除できます。

委任管理者を S3 コンソールを使用して登録解除するには、「S3 Storage Lens の委任された管理者の登録解除」を参照してください。

また、委任管理者は、管理アカウントから AWS Organizations の REST API、AWS CLI、または SDK を使用することでも登録解除できます。詳細については、AWS Organizations API リファレンスの「DeregisterDelegatedAdministrator」を参照してください。

> **Note**
>
> - また、委任管理者の登録を解除すると、委任管理者によって作成されたすべての組織レベルのダッシュボードで、ストレージのメトリクスの新たな集計が自動的に停止されます。
> - 委任管理者アカウントは、データがクエリに利用できる間は、それらのダッシュボードの履歴データを見ることができます。

Amazon S3 ストレージレンズアクセス許可

Amazon S3 ストレージレンズでは、S3 ストレージレンズの各アクションへのアクセスを許可するために、AWS Identity and Access Management (IAM) の新しいアクセス許可が必要となります。これらのアクセス許可を付与するには、アイデンティティベースの IAM ポリシーを使用できます。このポリシーを、IAM ユーザー、グループ、またはロールにアタッチして、アクセス許可を付与することができます。このようなアクセス許可には、S3 Storage Lens の有効化または無効化や、S3 Storage Lens ダッシュボードや設定へのアクセスなどが含まれます。

これらの IAM ユーザーまたはロールは、以下の条件の両方に当てはまる場合を除き、ダッシュボードまたはその設定を作成または所有しているアカウントに属している必要があります。

- アカウントが AWS Organizations のメンバーです。
- 委任管理者として管理アカウントによって、組織レベルのダッシュボードを作成するアクセスを付与されています。

> **Note**
>
> - Amazon S3 Storage Lens のダッシュボードを表示するために、アカウントのルートユーザーの認証情報を使用することはできません。S3 ストレージレンズダッシュボードにアクセスするには、新規または既存の IAM ユーザーに対し、必要な IAM アクセス許可を付与する必要があります。その後、それらのユーザーの認証情報によりサインインを行い、S3 Storage Lens ダッシュボードにアクセスします。詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。
> - Amazon S3 コンソールで S3 ストレージレンズを使用するには、複数のアクセス許可が必要になることがあります。例えば、コンソールでダッシュボードを編集するには、次の許可が必要です。
>   - `s3:ListStorageLensConfigurations`
>   - `s3:GetStorageLensConfiguration`
>   - `s3:PutStorageLensConfiguration`

トピック

- アカウントで S3 ストレージレンズを使用するためのアクセス許可の設定
- アカウントで S3 Storage Lens グループを使用するためのアクセス許可の設定

- AWS Organizations を使用した S3 ストレージレンズを使用するための許可の設定

アカウントで S3 ストレージレンズを使用するためのアクセス許可の設定

S3 Storage Lens ダッシュボードと Storage Lens ダッシュボード設定を作成して管理するには、実行するアクションに応じて次の権限が必要です。

Amazon S3 ストレージレンズ関連の IAM アクセス許可

アクション	IAM アクセス許可
Amazon S3 コンソールで S3 ストレージレンズダッシュボードを作成または更新します。	`s3:ListStorageLensConfigurations` `s3:GetStorageLensConfiguration` `s3:GetStorageLensConfigurationTagging` `s3:PutStorageLensConfiguration` `s3:PutStorageLensConfigurationTagging`
Amazon S3 コンソールで S3 ストレージレンズダッシュボードのタグを取得します。	`s3:ListStorageLensConfigurations` `s3:GetStorageLensConfigurationTagging`
Amazon S3 コンソールに S3 ストレージレンズダッシュボードを表示します。	`s3:ListStorageLensConfigurations` `s3:GetStorageLensConfiguration` `s3:GetStorageLensDashboard`
Amazon S3 コンソールで S3 ストレージレンズダッシュボードを削除します。	`s3:ListStorageLensConfigurations` `s3:GetStorageLensConfiguration` `s3:DeleteStorageLensConfiguration`

アクション	IAM アクセス許可
AWS CLI または AWS SDK により S3 ストレージレンズの設定を作成または更新します。	`s3:PutStorageLensConfiguration` `s3:PutStorageLensConfigurationTagging`
AWS CLI または AWS SDK により S3 ストレージレンズの設定のタグを取得します。	`s3:GetStorageLensConfigurationTagging`
AWS CLI または AWS SDK により S3 ストレージレンズの設定を表示します。	`s3:GetStorageLensConfiguration`
AWS CLI または AWS SDK により S3 ストレージレンズの設定を削除します。	`s3:DeleteStorageLensConfiguration`

Note

- IAM ポリシーでリソースタグを使用すると、アクセス許可を管理できます。
- これらのアクセス許可を持つ IAM ユーザーまたはロールは、バケットおよびプレフィックスからのメトリクスを参照できます。ただし、オブジェクトの読み出しまたは一覧表示のための、直接的なアクセス許可がない場合があります。
- プレフィックスレベルのメトリクスが有効になっている S3 Storage Lens ダッシュボードでは、選択したプレフィックスパスがオブジェクトキーと一致すると、ダッシュボードにそのオブジェクトキーが別のプレフィックスとして表示されることがあります。
- ご自身のアカウントのバケットに保存されているメトリクスをエクスポートする場合は、IAM ポリシー内の既存の `s3:GetObject` アクセス許可により権限が付与されます。これと同様に、AWS Organizations エンティティであれば、組織の管理アカウントまたは委任管理者アカウントとして IAM ポリシーを使用することで、組織レベルのダッシュボードと設定への許可を管理できます。

アカウントで S3 Storage Lens グループを使用するためのアクセス許可の設定

S3 Storage Lens グループを使用すると、プレフィックス、サフィックス、オブジェクトタグ、オブジェクトサイズ、またはオブジェクト経過時間に基づいてバケット内のストレージの分布を把握でき

ます。Storage Lens グループをダッシュボードにアタッチすると、集計されたメトリクススを表示できます。

Storage Lens グループを操作するには、特定の権限が必要です。詳細については、「the section called “Storage Lens グループのアクセス許可。”」を参照してください。

AWS Organizations を使用した S3 ストレージレンズを使用するための許可の設定

Amazon S3 ストレージレンズを使用することで、AWS Organizations の階層に属しているすべてのアカウントから、ストレージのメトリクスと使用状況に関するデータを収集できます。Organizations での S3 ストレージレンズの使用に関連する、アクションとアクセス許可を次に示します。

S3 ストレージレンズを使用するための AWS Organizations 関連 IAM 許可

アクション	IAM アクセス許可
S3ストレージレンズのために、信頼されたアクセスを組織内で有効にします。	`organizations:EnableAWSServiceAccess`
組織内で、S3 ストレージレンズのために信頼されたアクセスを無効にします。	`organizations:DisableAWSServiceAccess`
組織内で、S3 ストレージレンズダッシュボードまたはその設定を作成するために、委任管理者を登録します。	`organizations:RegisterDelegatedAdministrator`
委任管理者の登録を解除して、組織の S3 ストレージレンズダッシュボードまたは設定を作成できないようにします。	`organizations:DeregisterDelegatedAdministrator`
組織全体のために S3 ストレージレンズの設定を作成するための追加のアクセス許可	`organizations:DescribeOrganization` `organizations:ListAccounts` `organizations:ListAWSServiceAccessForOrganization`

アクション	IAM アクセス許可
	`organizations:ListDelegated Administrators` `iam:CreateServiceLinkedRole`

Amazon S3 ストレージレンズでのメトリクスの表示

S3 ストレージレンズはメトリクスを集約し、Amazon S3 コンソールの [Buckets] (バケット) ページの [Account snapshot] (アカウントのスナップショット) セクションにこの情報を表示します。S3 Storage Lens は、インサイトと傾向を可視化したり、外れ値にフラグ付けしたり、ストレージコストの最適化やデータ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取るために使用できるインタラクティブダッシュボードも提供します。ダッシュボードには、組織、アカウント、AWS リージョン、バケット、オブジェクト、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成して可視化できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。

デフォルトで、すべてのダッシュボードには無料のメトリクスが設定されています。メトリクスには、S3 ストレージ全体の使用状況とアクティビティを把握し、ストレージコストを最適化し、データ保護とアクセス管理のベストプラクティスを実装するために使用できるメトリクスが含まれています。無料のメトリクスはバケットレベルまで下って集計されます。データは、最長 14 日間のクエリで利用可能です。

高度なメトリクスとレコメンデーションには、ストレージ全体の使用状況やアクティビティを詳細に把握するために使用できる以下の追加機能や、ストレージを最適化するためのベストプラクティスが含まれています。

- コンテキストレコメンデーション (ダッシュボードでのみ使用可能)
- 高度なメトリクス (バケットごとに集計されたアクティビティメトリクスを含む)
- プレフィックス集約
- Storage Lens グループ集計
- Storage Lens グループ集計
- Amazon CloudWatch パブリッシング

高度なメトリクスデータは 15 か月間クエリで使用できます。S3 ストレージレンズで高度なメトリクスを使用するには、追加料金がかかります。詳細については、「Amazon S3 の料金」を参照してください。無料のメトリクスと高度なメトリクスの詳細については、「メトリクスの選択」を参照してください。

トピック

- ダッシュボードで S3 Storage Lens のメトリクスを表示する
- データエクスポートで Amazon S3 Storage Lens のメトリクスを確認する
- CloudWatch で S3 Storage Lens のメトリクスをモニタリング

ダッシュボードで S3 Storage Lens のメトリクスを表示する

Amazon S3 コンソールでは、S3 Storage Lens により提供されるインタラクティブなデフォルトダッシュボードが利用でき、これを使用してデータのインサイトと傾向を可視化できます。このダッシュボードでは、外れ値にフラグ付けして、ストレージコストの最適化や、データ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取ることもできます。ダッシュボードには、アカウント、バケット、AWS リージョン、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成できる、ドリルダウンオプションが用意されています。S3 Storage Lens が AWS Organizations と連携できるようにした場合、組織レベルでインサイトを生成することもできます (AWS Organizations ヒエラルキーに属するすべてのアカウントのデータなど)。ダッシュボードでは常に、メトリクスの集計が可能な、最新の日付で読み込みが行われます。

コンソールの S3 ストレージレンズのデフォルトダッシュボードの名前は、default-account-dashboard です。Amazon S3 ではこのダッシュボードが事前定義されており、アカウント全体の要約されたインサイトとトレンドを視覚化し、S3 コンソールで毎日更新します。デフォルトのダッシュボードのスコープ設定を変更することはできませんが、選択するメトリクスを、無料メトリクスから有料の高度な推奨のメトリクスに、アップグレードすることはできます。高度なメトリクスとレコメンデーションにより、その他のメトリクスや機能にアクセスできます。これらの機能には、高度なメトリクスカテゴリ、プレフィックスレベルでの集約、コンテキストに応じたレコメンデーション、Amazon CloudWatch パブリッシングなどがあります。

デフォルトのダッシュボードは無効にできますが、削除はできません。デフォルトのダッシュボードを無効にすると、更新されなくなります。S3 Storage Lens または S3 [Buckets] (バケット) ページのアカウントスナップショットで、新しい毎日のメトリクスを受け取ることはなくなります。データクエリの 14 日の有効期限が切れるまでは、デフォルトダッシュボードで履歴データを引き続き表示できます。高度なメトリクスとレコメンデーションを有効にしている場合、この期間は 15 か月です。このデータにアクセスするには、有効期限内にダッシュボードを再度有効化します。

追加の S3 Storage Lens ダッシュボードを作成し、、S3 バケット、またはアカウント (の場合) によってスコープを設定できます。Storage Lens が AWS Organizations と連携できるようにしている場合は、組織別にダッシュボードの範囲を設定することもできます。S3 Storage Lens ダッシュボードを作成または編集するとき、ダッシュボードのスコープとメトリクスの選択を定義します。

作成したその他のダッシュボードは無効化または削除できます。

- ダッシュボードを無効にすると、そのダッシュボードは更新されなくなるので、日次で新しいメトリクスが生成されることもなくなります。無料メトリクスの履歴データについては、14 日間の有効期限まで参照することが可能です。そのダッシュボードで高度なメトリクスとレコメンデーションを有効にした場合、この期間は 15 か月です。有効期限内であれば、ダッシュボードを再度有効化することで、このデータにアクセスできるようになります。
- ダッシュボードを削除すると、それまでの設定内容はすべて失われます。削除されたダッシュボードからは、それ以後の新しい日次のメトリクスは受信できなくなり、そのダッシュボードに関連付けられた履歴データにもアクセスできなくなります。削除したダッシュボードの履歴データにアクセスする場合は、同じホームリージョンで同じ名前を使用しながら、新たなダッシュボードを作成する必要があります。

トピック

Amazon S3 Storage Lens ダッシュボードの表示

以下の手順は、S3 コンソールで S3 ストレージレンズダッシュボードを表示する方法です。ダッシュボードを使用してコストを最適化する方法、ベストプラクティスを実装する方法、S3 バケットにアクセスするアプリケーションのパフォーマンスを向上させる方法を示すユースケースベースのチュートリアルについては、「Amazon S3 ストレージレンズメトリクスのユースケース」を参照してください。

> **Note**
>
> Amazon S3 Storage Lens のダッシュボードを表示するために、アカウントのルートユーザーの認証情報を使用することはできません。S3 ストレージレンズダッシュボードにアクセスするには、新規または既存の IAM ユーザーに対し、必要な AWS Identity and Access Management (IAM) アクセス許可を付与する必要があります。その後、それらのユーザー

の認証情報によりサインインを行い、S3 Storage Lens ダッシュボードにアクセスします。詳細については、IAM ユーザーガイドの「Amazon S3 ストレージレンズアクセス許可」と「IAM でのセキュリティベストプラクティス」を参照してください。

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。

   ダッシュボードは S3 ストレージレンズで開きます。[Snapshot for date] (日付のスナップショット) セクションには、S3 ストレージレンズがメトリクスを収集した最新の日付が表示されます。ダッシュボードでは常に、メトリクスの集計が可能な、最新の日付で読み込みが行われます。
4. (オプション) S3 ストレージレンズダッシュボードの日付を変更するには、右上の日付セレクターで新しい日付を選択します。
5. (オプション) ダッシュボードのデータの範囲をさらに限定するために一時的なフィルターを適用するには、次の操作を行います。

   a. [フィルター] セクションを展開します。

   b. 特定のアカウント、AWS リージョン、ストレージクラス、またはバケット、プレフィックス、または Strage Lens グループでフィルタリングするには、フィルタリングするオプションを選択します。

   > **Note**
   > プレフィックスフィルターと Storage Lens グループフィルターは同時に適用できません。

   c. フィルターを更新するには、[Apply] (適用) を選択します。

   d. フィルターを削除するには、フィルターの横にある X をクリックします。
6. S3 ストレージレンズダッシュボードのどのセクションでも、特定のメトリクスのデータを表示するには、[Metric] (メトリクス) でメトリクス名を選択します。
7. S3 Storage Lens ダッシュボードのどのグラフや可視化でも、[アカウント]、AWS リージョン、[ストレージクラス]、[バケット]、[プレフィックス]、または [Storage Lens グループ] タブを

使用して、より深いレベルの集計にドリルダウンできます。例については、「コールド Amazon S3 バケットを検出する」を参照してください。

S3 Storage Lens ダッシュボードを理解する

S3 ストレージレンズダッシュボードには、基本の [Overview] (概要) タブと、個別の集計レベルを表示するための最大 5 つの追加タブがあります。

- アカウント
- AWS リージョン
- ストレージクラス
- バケット
- プレフィックス
- Storage Lens グループ

[Overview] (概要) タブでは、ダッシュボードのデータが [Snapshot for date] (日付のスナップショット)、[Trends and distributions] (傾向とディストリビューション)、[Top N overview] (トップ N の概要) の 3 つのセクションに集約されます。

S3 ストレージレンズの詳細については、次のセクションを参照してください。

Snapshot

[Snapshot for date] (日付のスナップショット) セクションには、選択した日付で S3 ストレージレンズにより集計されたメトリクスの概要が表示されます。これらの要約メトリクスには、以下の指標が含まれます。

- 合計ストレージ — 使用されているストレージの合計容量 (バイト単位)。
- オブジェクト数 — AWS アカウント 内のオブジェクトの総数。
- 平均オブジェクトサイズ – オブジェクトの平均サイズ。
- アクティブなバケット – アカウントでストレージが 0 バイトを超える、アクティブに使用されているアクティブなバケットの総数。
- アカウント – ストレージがスコープ内にあるアカウントの数。AWS Organizations を使用している場合を除き、この値は 1 です。この場合、S3 ストレージレンズには、有効なサービスにリンクされたロールを持つ、信頼されたアクセス権があります。詳細については、「Amazon S3 ストレージレンズでのサービスにリンクされたロールの使用」を参照してください。

- バケット — アカウント内のバケットの総数。

メトリクスデータ

スナップショットに表示される各メトリクスについて、次のデータが表示されます。

- メトリクス名 – メトリクスの名前。
- メトリクスカテゴリー – メトリクスが分類されるカテゴリ。
- 日付の合計 – 選択した日付の合計数。
- % change – 最後のスナップショット日からの変化率。
- 30 日間のトレンド – 30 日間のメトリクスの変化を示すトレンドライン。
- レコメンデーション – スナップショットで提供されたデータに基づく状況に応じたレコメンデーション。レコメンデーションには、高度なメトリクスとレコメンデーションが用意されています。詳細については、「レコメンデーション」を参照してください。

メトリクスカテゴリ

必要に応じて、ダッシュボードの [Snapshot for date] (日付のスナップショット) セクションを更新して、他のカテゴリのメトリクスを表示できます。その他のメトリクスのスナップショットデータを表示したい場合は、以下の [Metrics categories] (メトリクスカテゴリ) から選択できます。

- コスト最適化
- データ保護
- アクティビティ (高度なメトリクスで利用可能)
- アクセス管理
- パフォーマンス
- のイベント

[Snapshot for date] (日付のスナップショット) セクションには、各カテゴリの選択したメトリクスのみが表示されます。特定のカテゴリのすべての指標を確認するには、[Trends and distributions] (傾向とディストリビューション)、[Top N overview] (トップ N の概要) セクションで指標を選択します。Storage Lens グループに メトリクスのカテゴリ リソースタグを追加する S3 ストレージレンズメトリクスの完全なリストについては、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

傾向と分布

[Overview] (概要) タブの 2 番目のセクションは、[Trends and distributions] (傾向とディストリビューション) です。[Trends and distributions] (傾向とディストリビューション) セクションでは、定義した日付範囲で比較するメトリクスを 2 つ選択できます。[Trends and distributions] (傾向とディストリビューション) セクションには、2 つのメトリクスの経時的な関係が表示されます。追跡している 2 つの傾向の間で、[Storage class] (ストレージクラス) と [Region] (リージョン) での分布を確認できます。オプションで、いずれかのチャートのデータポイントにドリルダウンして、より詳細な分析を行うことができます。

[Trends and distributions] (傾向とディストリビューション) セクションを使ったチュートリアルについては、「AWS KMS によるデフォルトの暗号化 (SSE-KMS) でサーバー側暗号化を使用しないバケットを識別する」を参照してください。

トップ N の概要

S3 Storage Lens ダッシュボードの 3 番目のセクションは、(昇順または降順でソートされた) トップ N の概要です。このセクションでは、AWS リージョン、バケット、プレフィックス、または Storage Lens グループのトップの数から選択したメトリクスが表示されます。S3 Storage Lens が AWS Organizations と連携できるようにしている場合、選択したメトリクスを表示することもできます。

[Top N overview] (トップ N の概要) セクションを使用するチュートリアルについては、「最大の S3 バケットを特定する」を参照してください。

オプションによるドリルダウンと次により分析

S3 Storage Lens ダッシュボードには、グラフ上の値を選択したときに表示される、アクションメニューが用意されており、これにより分析のための円滑な操作を実現しています。このメニューを使用するには、関連するメトリクス値を表示するには、グラフから任意の値を選択してから、表示されるボックスで次の 2 つのオプションから選択します。

- [Drill down] (ドリルダウン) アクションでは、選択した値が、ダッシュボードのすべてのタブでフィルターとして適用されます。その値にドリルダウンして、より深い分析を行うことができます。
- [次により分析:] アクションを実行すると、選択する [ディメンション] タブに移動し、そのタブ値をフィルターとして適用します。これらのタブには、[アカウント]、AWS リージョン、[ストレージクラス]、[バケット]、[プレフィックス] ([高度なメトリクス] と [プレフィックス集計] が有効になっているダッシュボード用)、および [Storage Lens グループ] ([高度なメトリクス] と [Storage Lens グループ集約] が有効になっているダッシュボード用) が含まれます。[次により分析:] を使う

と、データを新しいディメンションのコンテキストで表示しながら、より深い分析を行うことができます。

非論理的な結果が出力された場合、あるいは値が存在しない場合には、[ドリルダウン] と [次により分析:] アクションが無効化されることがあります。[ドリルダウン] と [次により分析:] アクションでは、ダッシュボードのすべてのタブにわたって、既存のフィルターの上に新たなフィルタが適用されます。必要に応じて、フィルタを削除することも可能です。

タブ

ディメンションレベルタブには、特定のディメンション内のすべての値に関する詳細が表示されます。例えば、[AWS リージョン] タブにはすべての AWS リージョン のメトリクスが表示され、[バケット] タブにはすべてのバケットのメトリクスが表示されます。各ディメンションタブは、次の 4 つのセクションからなる共通のレイアウトで構成されています。

- 選択したメトリクスのディメンション内で、過去 30 日間のトップ N 個の項目を表示する傾向グラフ。このグラフが表示する項目数は、デフォルトでトップ 10 個に設定されていますが、上位 3 個に減らしたり、50 個まで増やすことができます。
- 選択した日付とメトリクスについての縦棒グラフを表示する ヒストグラムグラフ。このグラフに表示されるアイテムの数が非常に多い場合は、水平方向にスクロールする必要がある場合があります。
- ディメンション内のすべての項目をプロットするバブル分析チャート。このチャートは X 軸に 1 番目のメトリクスを、Y 軸に 2 番目のメトリクスを表します。3 番目のメトリクスはバブルのサイズで表されます。
- 行内でリストされた、ディメンション内の各アイテムを表示する、メトリクスグリッドビュー。列には、利用可能なメトリクスがメトリクスカテゴリのタブで整理されて表示されるので、ナビゲーションが容易に行えます。

データエクスポートで Amazon S3 Storage Lens のメトリクスを確認する

Amazon S3 ストレージレンズのメトリクスは、CSV または Apache Parquet 形式を使用した、メトリクスのエクスポートファイルとして毎日生成され、アカウントの S3 バケットに保存されます。バケットにエクスポートされたメトリクスは、Amazon QuickSight や Amazon Athena などの任意の分析ツールに取り込むことで、ストレージの使用状況とアクティビティの傾向を分析できるようになります。

トピック

- AWS KMS key を使用してメトリクスのエクスポートを暗号化する
- S3 Storage Lens のエクスポートマニフェストとは?
- Amazon S3 Storage Lens のエクスポートスキーマを理解する

AWS KMS key を使用してメトリクスのエクスポートを暗号化する

カスタマーマネージドキーを使用してメトリクスのエクスポートを暗号化するアクセス許可を Amazon S3 ストレージレンズに付与するには、キーポリシーを使用する必要があります。S3 ストレージレンズのメトリクスのエクスポートの暗号化に KMS を使用できるようにキーポリシーを変更するには、次の手順に従います。

S3 ストレージレンズに KMS キーを使用したデータの暗号化のアクセス許可を付与するには

1. カスタマーマネージドキーを所有する AWS アカウント を使用して、AWS Management Console にサインインします。
2. AWS KMS コンソール (https://console.aws.amazon.com/kms) を開きます。
3. AWS リージョン を変更するには、ページの右上隅にあるリージョンセレクターを使用します。
4. 左のナビゲーションペインで、[Customer managed keys] を選択します。
5. [カスタマーマネージドキー] で、メトリクスのエクスポートを暗号化するために使用するキーを選択します。AWS KMS keys はリージョン固有で、メトリクスのエクスポート先である S3 バケットと同じリージョンにある必要があります。
6. [Key policy] 行で、[Switch to policy view] を選択します。
7. [編集] をクリックし、キーポリシーを更新します。
8. [キーポリシーの編集] で、既存のキーポリシーに次のキーポリシーを追加します。このポリシーを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
    "Sid": "Allow Amazon S3 Storage Lens use of the KMS key",
     "Effect": "Allow",
    "Principal": {
        "Service": "storage-lens.s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
```

```
        "StringEquals": {
            "aws:SourceArn": "arn:aws:s3:us-east-1:source-account-id:storage-
lens/your-dashboard-name",
            "aws:SourceAccount": "source-account-id"
         }
      }
}
```

9. [Save changes] (変更の保存) をクリックします。

カスタマーマネージドキーの作成とキーポリシーの使用の詳細については、AWS Key Management Service デベロッパーガイドの以下のトピックを参照してください。

- IAM の使用開始
- AWS KMS でのキーポリシーの使用

また、AWS KMS PUT キーポリシー API オペレーション (PutKeyPolicy) を使用して、カスタマーマネージドキーにキーポリシーをコピーすることもできます。このポリシーは、REST API、AWS CLI、および SDK を使用してメトリクスのエクスポートを暗号化する際に使用します。

S3 Storage Lens のエクスポートマニフェストとは?

集計されるデータが大量の場合、S3 Storage Lens の日次メトリクスのエクスポートは、複数のファイルに分割されます。マニフェストファイル `manifest.json` には、その日のメトリクスのエクスポートファイルが保存された場所が記述されています。新しいエクスポートが配信されるたびに、マニフェストも新たに添付されます。`manifest.json` ファイルに含まれる各マニフェストには、対象のエクスポートに関するメタデータおよびその他の基本的な情報が記載されています。

マニフェストの情報には、次のプロパティが含まれています。

- `sourceAccountId` – その設定の所有者であるアカウント ID。
- `configId` – ダッシュボードの一意の識別子。
- `destinationBucket` – メトリクスのエクスポートが保存されるバケットの、Amazon リソースネーム (ARN)。
- `reportVersion` – エクスポートのバージョン。
- `reportDate` – レポートの日付。
- `reportFormat` – レポートの形式。
- `reportSchema` – レポートのスキーマ。

- reportFiles – 保存先のバケットにある、実際のエクスポートレポートファイルのリスト。

CSV 形式で作成されたエクスポートの、manifest.json ファイルに記述されたマニフェストの例を次に示します。

```
{
   "sourceAccountId":"123456789012",
   "configId":"my-dashboard-configuration-id",
   "destinationBucket":"arn:aws:s3:::destination-bucket",
   "reportVersion":"V_1",
   "reportDate":"2020-11-03",
   "reportFormat":"CSV",

 "reportSchema":"version_number,configuration_id,report_date,aws_account_number,aws_region,stor
   "reportFiles":[
        {
            "key":"DestinationPrefix/StorageLens/123456789012/my-dashboard-
configuration-id/V_1/reports/dt=2020-11-03/a38f6bc4-2e3d-4355-ac8a-e2fdcf3de158.csv",
            "size":1603959,
            "md5Checksum":"2177e775870def72b8d84febe1ad3574"
        }
      ]
}
```

Parquet 形式のエクスポート用の manifest.json ファイルに含まれるマニフェストの例を次に示します。

```
{
   "sourceAccountId":"123456789012",
   "configId":"my-dashboard-configuration-id",
   "destinationBucket":"arn:aws:s3:::destination-bucket",
   "reportVersion":"V_1",
   "reportDate":"2020-11-03",
   "reportFormat":"Parquet",
   "reportSchema":"message s3.storage.lens { required string version_number;
 required string configuration_id; required string report_date; required string
 aws_account_number; required string aws_region; required string storage_class;
 required string record_type; required string record_value; required string
 bucket_name; required string metric_name; required long metric_value; }",
   "reportFiles":[
      {
```

```
         "key":"DestinationPrefix/StorageLens/123456789012/my-dashboard-configuration-
id/V_1/reports/dt=2020-11-03/bd23de7c-b46a-4cf4-bcc5-b21aac5be0f5.par",
         "size":14714,
         "md5Checksum":"b5c741ee0251cd99b90b3e8eff50b944"
      }
}
```

メトリクスのエクスポートの生成は、Amazon S3 コンソールでダッシュボード設定の一部とするか、Amazon S3 REST API、AWS CLI、および SDK を使用するかの、どちらかで実行できます。

Amazon S3 Storage Lens のエクスポートスキーマを理解する

次の表に、S3 Storage Lens でのメトリクスのエクスポートスキーマを示します。

Attribute Nameは	データ型	列名	説明
`VersionNumber`	文字列	`version_number`	使用中の S3 Storage Lens メトリクスのバージョン。
`ConfigurationId`	文字列	`configuration_id`	S3 ストレージレンズ設定の `configuration_id` 。
`ReportDate`	文字列	`report_date`	メトリクスが追跡された日付。
`AwsAccountNumber`	文字列	`aws_account_number`	お客様の AWS アカウント 番号。
`AwsRegion`	文字列	`aws_region`	メトリクスが追跡されている AWS リージョン。
`StorageClass`	文字列	`storage_class`	調査対象のバケットのストレージクラス。
`RecordType`	ENUM	`record_type`	レポートされているアーティファクトの

Attribute Nameは	データ型	列名	説明
			タイプ (ACCOUNT、BUCKET、もしくは PREFIX)。
`RecordValue`	文字列	`record_value`	`RecordType` アーティファクトの値。 **Note** `record_value` は、URL エンコードされています。
`BucketName`	文字列	`bucket_name`	レポートされているバケットの名前。
`MetricName`	文字列	`metric_name`	レポートされているメトリクスの名前。
`MetricValue`	Long	`metric_value`	レポートされているメトリクスの値。

S3 Storage Lens でのメトリクスのエクスポート例

次に示すのは、前出のスキーマに基づく S3 Storage Lens でのメトリクスのエクスポートの例です。

Note

Storage Lens グループのメトリクスは、`record_type` 列内の `STORAGE_LENS_GROUP_BUCKET` または `STORAGE_LENS_GROUP_ACCOUNT` 値を探すことで識別できます。`record_value` 列には、Storage Lens グループの Amazon リソースネーム (ARN) が表示されます (例: `arn:aws:s3:us-east-1:123456789012:storage-lens-group/slg-1`)。

version_r	configuration_id	report_date	aws_account_number	aws_region	storage_class	record_type	record_value	bucket_name	metric_name	metric_value
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			StorageBytes	2478830621
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectCount	1598962
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ReplicatedStorageBytes	20000
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ReplicatedObjectCount	20
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			EncryptedStorageBytes	2478828742
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			EncryptedObjectCount	1598961
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			DeleteMarkerObjectCount	1500
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectLockEnabledStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectLockEnabledObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			CurrentVersionStorageBytes	2478830621
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			CurrentVersionObjectCount	1598962
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			NonCurrentVersionStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			NonCurrentVersionObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			IncompleteMultipartUploadStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			IncompleteMultipartUploadObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	StorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ReplicatedStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ReplicatedObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	EncryptedStorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	EncryptedObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	DeleteMarkerObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ObjectLockEnabledStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ObjectLockEnabledObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	CurrentVersionStorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	CurrentVersionObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	NonCurrentVersionStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	NonCurrentVersionObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	IncompleteMultipartUploadStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	IncompleteMultipartUploadObjectCount	0

以下は、Storage Lens グループのデータを使った Storage Lens メトリクスのエクスポートの例です。

version_numbe	configuration_id	report_date	aws_account_num	aws_region	storage_class	record_type	record_value	bucket_name	metric_name	metric_value
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		StorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		EncryptedStorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		EncryptedObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		DeleteMarkerObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectLockEnabledStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectLockEnabledObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		CurrentVersionStorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		CurrentVersionObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		NonCurrentVersionStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		NonCurrentVersionObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMultipartUploadStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMultipartUploadObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		DeleteMarkerStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedStorageBytesSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedObjectCountSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMPUStorageBytesOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMPUObjectCountOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	StorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	EncryptedStorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	EncryptedObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	DeleteMarkerObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectLockEnabledStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectLockEnabledObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	CurrentVersionStorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	CurrentVersionObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	NonCurrentVersionStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	NonCurrentVersionObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMultipartUploadStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMultipartUploadObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	DeleteMarkerStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedStorageBytesSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedObjectCountSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMPUStorageBytesOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMPUObjectCountOlderThan7Days	0

CloudWatch で S3 Storage Lens のメトリクスをモニタリング

S3 ストレージレンズのメトリクスを Amazon CloudWatch に公開すると、CloudWatch ダッシュボードで運用状況を一元的に表示できます。また、アラームやトリガーアクション、メトリクス計算、異常検出などの CloudWatch 機能を使用して、S3 ストレージレンズのメトリクスをモニタリングしてアクションを実行できます。さらに CloudWatch API オペレーションにより、サードパーティープロバイダーを含むアプリケーションが S3 ストレージレンズのメトリクスにアクセスできる

ようになります。CloudWatch Events の詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

Amazon S3 コンソール、Amazon S3 REST API、AWS CLI および AWS SDK を使用して、新規または既存のダッシュボードの設定で CloudWatch 公開オプションを有効にできます。S3 Storage Lens のアドバンストメトリクスとレコメンデーションにアップグレードされたダッシュボードは、CloudWatch の公開オプションを使用できます。S3 ストレージレンズの高度なメトリクスとレコメンデーションの料金については、「Amazon S3 料金表」を参照してください。追加の CloudWatch メトリクスの公開料金は適用されません。ただし、ダッシュボード、アラーム、API コールなどのその他の CloudWatch 料金が適用されます。詳細については、Amazon CloudWatch の料金を参照してください。

S3 Storage Lens のメトリクスは、S3 Storage Lens の設定を所有するアカウント内の CloudWatch に公開されます。高度なメトリクスとレコメンデーションで CloudWatch 公開オプションを有効にすると、CloudWatch の組織、アカウント、バケットレベルのメトリクスにアクセスできます。プレフィックスレベルのメトリクスは CloudWatch では使用できません。

> **Note**
>
> S3 Storage Lens メトリクスは毎日のメトリクスであり、CloudWatch に 1 日 1 回公開されます。CloudWatch で S3 Storage Lens メトリクスをクエリする場合、クエリの期間は 1 日 (86400 秒) にする必要があります。毎日の S3 Storage Lens メトリクスが Amazon S3 コンソールの S3 Storage Lens ダッシュボードに表示された後、これらの同じメトリクスが CloudWatch に表示されるには数時間かかることがあります。初めて S3 Storage Lens メトリクスの CloudWatch 公開オプションを有効にすると、メトリクスが CloudWatch に公開されるまで最大で 24 時間かかることがあります。

CloudWatch 公開オプションを有効にすると、次の CloudWatch 機能を使用して S3 ストレージ lensStorage レンズデータをモニタリングおよび分析できます。

- ダッシュボード - CloudWatch ダッシュボードを使用して、カスタマイズされた S3 Storage Lens ダッシュボードを作成します。CloudWatch ダッシュボードを AWS アカウント に直接アクセスできないユーザー (チーム間、ステークホルダー、組織外の人) と共有します。
- アラームとトリガーされたアクション - メトリクスを監視し、閾値に違反したときにアクションを起こすアラームを構成することができます。例えば、未完了のマルチパートアップロードバイトメトリクスが 3 日間連続で 1 GB を超えた場合に Amazon SNS 通知を送信するアラームを設定できます。

- 異常検出 - 異常検出を有効にして、メトリクスを継続的に分析し、正常なベースラインを決定後、異常を表面化させます。メトリクスの想定値に基づいて異常検出アラームを作成できます。例えば、オブジェクトロックが有効になっているバイト数メトリクスに異常がないかをモニタリングすると、オブジェクトロック設定の不正な削除を検出できます。
- Metric Math - これにより複数の S3 Storage Lens メトリクスをクエリし、数式を使用して、これらのメトリクスに基づく新しい時系列を作成できます。例えば、`StorageBytes` を `ObjectCount` で割って、平均オブジェクトサイズを取得するための新しいメトリクスを作成できます。

S3 Storage Lens メトリクスの CloudWatch 公開オプションの詳細については、以下のトピックを参照してください。

トピック

- S3 Storage Lens のメトリクスとディメンション
- S3 Storage Lens 用の CloudWatch パブリッシングの有効化
- CloudWatch で S3 Storage Lens のメトリクスを表示します。

S3 Storage Lens のメトリクスとディメンション

S3 Storage Lens メトリクスを CloudWatch に送信するには、S3 Storage Lens のアドバンストメトリクスとレコメンデーションにある CloudWatch 公開オプションを有効にする必要があります。高度なメトリクスを有効にすると、CloudWatch ダッシュボードを使用して、S3 ストレージレンズのメトリクスを他のアプリケーションメトリクスとともにモニタリングし、運用状態の統一ビューを作成できます。ディメンションを使用して CloudWatch の S3 Storage Lens メトリクスを組織、アカウント、バケット、ストレージクラス、リージョン、メトリクスの設定 ID でフィルタリングできます。

S3 Storage Lens のメトリクスは、S3 Storage Lens の設定を所有するアカウント内の CloudWatch に公開されます。高度なメトリクスとレコメンデーションで CloudWatch 公開オプションを有効にすると、CloudWatch の組織、アカウント、バケットレベルのメトリクスにアクセスできます。プレフィックスレベルのメトリクスは CloudWatch では使用できません。

> **Note**
>
> S3 Storage Lens メトリクスは毎日のメトリクスであり、CloudWatch に 1 日 1 回公開されます。CloudWatch で S3 Storage Lens メトリクスをクエリする場合、クエリの期間は 1 日 (86400 秒) にする必要があります。毎日の S3 Storage Lens メトリクスが Amazon S3 コ

ンソールの S3 Storage Lens ダッシュボードに表示された後、これらの同じメトリクスが CloudWatch に表示されるには数時間かかることがあります。初めて S3 Storage Lens メトリクスの CloudWatch 公開オプションを有効にすると、メトリクスが CloudWatch に公開されるまで最大で 24 時間かかることがあります。

CloudWatch での S3 Storage Lens のメトリクスとディメンションの詳細については、以下のトピックを参照してください。

トピック

- メトリクス
- ディメンション

メトリクス

S3 ストレージレンズのメトリクスは、CloudWatch 内でメトリクスとして使用できます。S3 Storage Lens のメトリクスは、`AWS/S3/Storage-Lens` 名前空間に公開されます。この名前空間は S3 Storage Lens のメトリクス専用です。Amazon S3 バケット、リクエスト、レプリケーションのメトリックスは、`AWS/S3` 名前空間に公開されます。

S3 Storage Lens のメトリクスは、S3 Storage Lens の設定を所有するアカウント内の CloudWatch に公開されます。高度なメトリクスとレコメンデーションで CloudWatch 公開オプションを有効にすると、CloudWatch の組織、アカウント、バケットレベルのメトリクスにアクセスできます。プレフィックスレベルのメトリクスは CloudWatch では使用できません。

S3 Storage Lens では、メトリクスが集計され、指定されたホームリージョンにのみ保存されます。S3 Storage Lens の設定で指定した、ホームリージョン内の CloudWatch にも S3 Storage Lens メトリクスが公開されます。

CloudWatch で使用可能なメトリックスのリストを含む S3 Storage Lens メトリクスの完全なリストについては、Amazon S3 Storage Lens のメトリクスに関する用語集 を参照してください。

> **Note**
>
> CloudWatch での S3 Storage Lens メトリクスの有効な統計情報は、Average です。CloudWatch での統計の詳細については、Amazon CloudWatch ユーザーガイドの CloudWatch 統計定義を参照してください。

CloudWatch での S3 Storage Lens のメトリクスの粒度について

S3 Storage Lens は、組織、アカウント、バケット、プレフィックスの粒度のメトリクスを提供します。S3 Storage Lens は、組織、アカウント、バケットレベルの S3 Storage Lens メトリクスを CloudWatch に公開します。プレフィクスレベルの S3 Storage Lens メトリクスは CloudWatch では使用できません。

CloudWatch で使用できる S3 Storage Lens メトリクスの粒度の詳細については、以下のリストを参照してください。

- 組織 - 組織内のメンバーアカウント全体で集計されたメトリクス。S3 Storage Lens は、メンバーアカウントのメトリクスを管理アカウントの CloudWatch に公開します。
  - 組織とアカウント - 組織のメンバーアカウントのメトリクス。
  - 組織とバケット - 組織のメンバーアカウントの Amazon S3 バケットのメトリクス。
- アカウント (組織レベル以外) - アカウントのバケット全体で集計されたメトリクス。
- バケット (組織レベル以外) - 特定のバケットのメトリクス。CloudWatch では、 S3 Storage Lens はこれらのメトリクスを、 S3 Storage Lens の設定を作成した AWS アカウント に公開します。S3 Storage Lens は、組織外の構成に対してのみ、これらのメトリクスを公開します。

ディメンション

S3 ストレージレンズで CloudWatch にデータが送信される際、ディメンションが各メトリクスにアタッチされます。ディメンションは、メトリクスの特性を記述するカテゴリです。CloudWatch が返す結果にフィルタを掛ける際にディメンションを使用できます。

例えば、 CloudWatch のすべての S3 Storage Lens のメトリクスは `configuration_id` ディメンションを持っています。このディメンションを使用して、特定の S3 Storage Lens 設定に関連付けられているメトリクスを区別できます。`organization_id` は、組織レベルのメトリクスを示します。CloudWatch のディメンションの詳細については、「CloudWatch ユーザーガイド 」の「ディメンション」を参照してください。

S3 ストレージレンズのメトリクスでは、メトリクスの粒度に応じて異なるディメンションを使用できます。例えば、`organization_id` ディメンションを使用して、組織レベルのメトリクスを AWS Organizations ID でフィルタリングすることができます。ただし、このディメンションをバケットおよびアカウントレベルのメトリクスに使用することはできません。詳細については、「ディメンションを使用した指標のフィルタリング」を参照してください。

S3 Storage Lens 構成で使用できるディメンションを確認するには、次の表を参照してください。

ディメンション	説明	バケット	アカウント	組織	組織とバケット	組織とアカウント
configuration_id	メトリクスでレポートされた S3 ストレージレンズ設定のダッシュボード名	.	.	.	.	.
metrics_version	S3 ストレージレンズのメトリクスのバージョン このメトリクスのバージョンには固定値 1.0 が含まれます。	.	.	.	.	.
organization_id	メトリクスの AWS Organizations ID	.	.	.	.	.
aws_account_number	メトリクスに関連付けられた AWS アカウント。	.	.	.	.	.
aws_region	メトリクスの AWS リージョン	.	.	.	.	.
bucket_name	メトリクスでレポートされた S3 バケットの名前	.	.	.	.	.
storage_class	メトリクスでレポートされたバケットのストレージクラス	.	.	.	.	.
record_type	メトリクスの粒度: 組織、アカウント、バケット	. バケット	. アカウント	. バケット	. アカウント	. 組織

S3 Storage Lens 用の CloudWatch パブリッシングの有効化

S3 ストレージレンズのメトリクスを Amazon CloudWatch に公開すると、CloudWatch ダッシュボードで運用状況を一元的に表示できます。また、アラームやトリガーアクション、メトリクス計算、異常検出などの CloudWatch 機能を使用して、S3 ストレージレンズのメトリクスをモニタリングしてアクションを実行できます。さらに CloudWatch API オペレーションにより、サードパーティープロバイダーを含むアプリケーションが S3 ストレージレンズのメトリクスにアクセスできるようになります。CloudWatch Events の詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

S3 Storage Lens のメトリクスは、S3 Storage Lens の設定を所有するアカウント内の CloudWatch に公開されます。高度なメトリクスとレコメンデーションで CloudWatch 公開オプションを有効にすると、CloudWatch の組織、アカウント、バケットレベルのメトリクスにアクセスできます。プレフィックスレベルのメトリクスは CloudWatch では使用できません。

S3 コンソール、Amazon S3 REST API、AWS CLI および AWS SDK を使用して、新規または既存のダッシュボード構成で CloudWatch サポートを有効にすることができます。CloudWatch の公開オプションは、S3 ストレージレンズの高度なメトリクスとレコメンデーションにアップグレードされたダッシュボードで利用可能です。S3 ストレージレンズの高度なメトリクスとレコメンデーションの料金については、「Amazon S3 料金表」を参照してください。追加の CloudWatch メトリクスの公開料金は適用されません。ただし、ダッシュボード、アラーム、API コールなどのその他の CloudWatch 料金が適用されます。

S3 Storage Lens メトリクスの CloudWatch パブリッシングオプションを有効にするには、以下のトピックを参照してください。

> **Note**
>
> S3 Storage Lens メトリクスは毎日のメトリクスであり、CloudWatch に 1 日 1 回公開されます。CloudWatch で S3 Storage Lens メトリクスをクエリする場合、クエリの期間は 1 日 (86400 秒) にする必要があります。毎日の S3 Storage Lens メトリクスが Amazon S3 コンソールの S3 Storage Lens ダッシュボードに表示された後、これらの同じメトリクスが CloudWatch に表示されるには数時間かかることがあります。初めて S3 Storage Lens メトリクスの CloudWatch 公開オプションを有効にすると、メトリクスが CloudWatch に公開されるまで最大で 24 時間かかることがあります。
>
> 現在、S3 ストレージレンズのメトリクスは CloudWatch ストリームを介して使用することはできません。

S3 コンソールの使用

S3 ストレージレンズのダッシュボードを更新するとき、ダッシュボード名、またはホームリージョンを変更することはできません。また、デフォルトダッシュボードのスコープを変更することもできません。これは、アカウント全体のストレージにスコープされています。

S3 ストレージレンズのダッシュボードを更新して CloudWatch パブリッシングを有効にするには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[S3 Storage Lens] (S3 ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. 編集するダッシュボードを選択し、 [編集] をクリックします。
4. メトリクスの選択で [Advanced metrics and recommendations] (アドバンストメトリクスとレコメンデーション) を選択します。

   アドバンストメトリクスとレコメンデーションには追加料金が適用されます。高度なメトリクスとレコメンデーションには、データクエリの 15 か月間、プレフィックスレベルで集計された使用状況メトリクス、バケットごとに集計されたアクティビティメトリクス、CloudWatch 公開オプション、ストレージコストの最適化とデータ保護のベストプラクティスの適用に役立つコンテキストに基づいた推奨事項が含まれています。詳細については、「Amazon S3 の料金」を参照してください。
5. [Select Advanced metrics and recommendations features] (高度なメトリクスとレコメンデーションを選択) から [CloudWatch publishing] (CloudWatch パブリッシング) を選択します。

   > **Important**
   >
   > 設定で使用状況メトリクスのプレフィクス集約を有効にすると、プレフィックスレベルのメトリックスは CloudWatch に公開されません。バケット、アカウント、組織レベルの S3 Storage Lens メトリクスのみが CloudWatch に公開されます。
6. [Save changes] (変更の保存) をクリックします。

CloudWatch サポートを有効にする新しい S3 Storage Lens ダッシュボードを作成するには、次のようにします。

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。

2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [ダッシュボードの作成] を選択します。
4. [General] (全般) で、次の設定オプションを定義します。

   a. [Dashboard name] (ダッシュボード名) に、ダッシュボード名を入力します。

      ダッシュボード名は 65 文字未満で記述する必要があり、特殊文字またはスペースを含めることはできません。ダッシュボードの作成後にダッシュボード名を変更することはできません。

   b. [ホームリージョン] から、ダッシュボードに対応するものを選択します。

      このダッシュボードスコープに含まれるすべてのリージョンのメトリクスは、指定されたホームリージョンに集約され保存されます。CloudWatch を使用した S3 Storage Lens メトリクスはホームリージョンでも利用できます。ダッシュボードの作成後にホームリージョンを変更することはできません。

5. (オプション) タグを追加するには、[タグを追加] を選択し、そのタグの キー と 値 を入力します。

   > **Note**
   >
   > ダッシュボードの設定には、最大 50 個のタグを追加できます。

6. 設定のスコープを定義します。

   a. 組織レベルの設定を作成する場合は、[Include all accounts in your configuration] (設定にすべてのアカウントを含める) または [Limit the scope to your signed-in account] (サインインしたアカウントに範囲を制限する) のいずれかの設定に含めるアカウントを選択します。

      > **Note**
      >
      > すべてのアカウントを含める組織レベルの設定を作成する場合、含めたり、除外したりできるのはリージョンのみで、バケットはできません。

   b. S3 ストレージレンズをダッシュボード構成に含めるリージョンとバケットを選択するには、以下の手順を実行します。

- すべてのリージョンを含めるには、[Include Regions and buckets] (リージョンとバケットを含める) を選択します。
- 特定のリージョンを含めるには、[Include all Regions] (すべてのリージョンを含める) をクリアにします。[Choose Regions to include] (含めるリージョンを選択) で、S3 Storage Lens がダッシュボードに含めるリージョンを選択します。
- 特定のバケットを含めるには、[Include all buckets] (すべてのバケットを含める) をクリアにします。[Choose buckets to include] (含めるバケットを選択) で、 S3 Storage Lens がダッシュボードに含めるバケットを選択します。

> **Note**
>
> バケットは最大 50 個まで選択できます。

7. [Metrics selection] (メトリクスの選択) で [Advanced metrics and recommendations] (高度なメトリクスとレコメンデーション) を選択します。

   高度なメトリクスとレコメンデーションの料金設定の詳細については、[Amazon S3 pricing] (Amazon S3 の料金) を参照してください。。

8. [Advanced metrics and recommendations features] (高度なメトリクスとレコメンデーションの機能) で、有効にするオプションを選択します。

   - Advanced metrics (高度なメトリクス)
   - [CloudWatch publishing] (CloudWatch パブリッシング)

   > **Important**
   >
   > S3 Storage Lens 設定でプレフィックス集約を有効にすると、プレフィックスレベルのメトリクスは CloudWatch に公開されません。バケット、アカウント、組織レベルの S3 Storage Lens メトリクスのみが CloudWatch に公開されます。

   - プレフィックス集約

> **Note**
>
> 高度なメトリクスとレコメンデーションの機能の詳細については、「メトリクスの選択」を参照してください。

9. [Advanced metrics] (高度なメトリクス) を有効にした場合は、S3 ストレージレンズのダッシュボードに表示する [Advanced metrics categories] (高度なメトリクスのカテゴリ) を選択します。

   - アクティビティのメトリクス
   - Detailed status code metrics (詳細なステータスコードのメトリクス)
   - Advanced cost optimization metrics (高度なコスト最適化メトリクス)
   - Advanced data protection metrics (高度なデータ保護メトリクス)

   メトリクスのカテゴリの詳細については、「メトリクスのカテゴリ」を参照してください。メトリクスの一覧については、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。
10. (オプション) メトリクスのエクスポートを設定します。

    ウィザードの各ステップの設定方法については、「Amazon S3 Storage Lens ダッシュボードの作成」を参照してください。
11. [ダッシュボードの作成] を選択します。

AWS CLI を使用する場合

次の AWS CLI の例では、S3 ストレージレンズの組織レベルの高度なメトリクスとレコメンデーションの構成を使用して、CloudWatch の公開オプションを有効にします。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control put-storage-lens-configuration --account-id=555555555555 --config-
id=your-configuration-id --region=us-east-1 --storage-lens-configuration=file://./
config.json

config.json
{
  "Id": "SampleS3StorageLensConfiguration", //Use this property to identify your S3
 Storage Lens configuration.
  "AwsOrg": { //Use this property when enabling S3 Storage Lens for AWS Organizations.
    "Arn": "arn:aws:organizations::123456789012:organization/o-abcdefgh"
  },
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
```

```
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "DetailedStatusCodesMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "ActivityMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "AdvancedCostOptimizationMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "DetailedStatusCodesMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "PrefixLevel":{
        "StorageMetrics":{
          "IsEnabled":true, //Mark this as false if you want only free metrics.
          "SelectionCriteria":{
            "MaxDepth":5,
            "MinStorageBytesPercentage":1.25,
            "Delimiter":"/"
          }
        }
      }
    }
  },
  "Exclude": { //Replace with "Include" if you prefer to include Regions.
    "Regions": [
      "eu-west-1"
    ],
    "Buckets": [ //This attribute is not supported for AWS Organizations-level
configurations.
      "arn:aws:s3:::source_bucket1"
    ]
  },
  "IsEnabled": true, //Whether the configuration is enabled
  "DataExport": { //Details about the metrics export
```

```
    "S3BucketDestination": {
      "OutputSchemaVersion": "V_1",
      "Format": "CSV", //You can add "Parquet" if you prefer.
      "AccountId": "111122223333",
      "Arn": "arn:aws:s3:::destination-bucket-name", // The destination bucket for your
 metrics export must be in the same Region as your S3 Storage Lens configuration.
      "Prefix": "prefix-for-your-export-destination",
      "Encryption": {
        "SSES3": {}
      }
    },
    "CloudWatchMetrics": {
      "IsEnabled": true //Mark this as false if you want to export only free metrics.
    }
  }
}
```

AWS SDK for Java の使用

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.CloudWatchMetrics;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
```

```
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateAndUpdateDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        String exportAccountId = "Destination Account ID";
        String exportBucketArn = "arn:aws:s3:::destBucketName"; // The destination
 bucket for your metrics export must be in the same Region as your S3 Storage Lens
 configuration.
        String awsOrgARN = "arn:aws:organizations::123456789012:organization/o-
abcdefgh";
        Format exportFormat = Format.CSV;

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
 AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
 DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
 PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
```

```
                    .withAdvancedDataProtectionMetrics(new
AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            Include include = new Include()
                    .withBuckets(Arrays.asList("arn:aws:s3:::bucketName"))
                    .withRegions(Arrays.asList("us-west-2"));

            StorageLensDataExportEncryption exportEncryption = new
StorageLensDataExportEncryption()
                    .withSSES3(new SSES3());
            S3BucketDestination s3BucketDestination = new S3BucketDestination()
                    .withAccountId(exportAccountId)
                    .withArn(exportBucketArn)
                    .withEncryption(exportEncryption)
                    .withFormat(exportFormat)
                    .withOutputSchemaVersion(OutputSchemaVersion.V_1)
                    .withPrefix("Prefix");
            CloudWatchMetrics cloudWatchMetrics = new CloudWatchMetrics()
                    .withIsEnabled(true);
            StorageLensDataExport dataExport = new StorageLensDataExport()
                    .withCloudWatchMetrics(cloudWatchMetrics)
                    .withS3BucketDestination(s3BucketDestination);

            StorageLensAwsOrg awsOrg = new StorageLensAwsOrg()
                    .withArn(awsOrgARN);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withInclude(include)
                    .withDataExport(dataExport)
                    .withAwsOrg(awsOrg)
                    .withIsEnabled(true);

            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
```

```
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

REST API の使用

Amazon S3 REST API を使用して CloudWatch 公開オプションを有効にするには、PutStorageLensConfiguration を使用します。

次のステップ

CloudWatch 公開オプションを有効にすると、 CloudWatch の S3 Storage Lens メトリクスにアクセスできます。CloudWatch 機能を活用して、CloudWatch で S3 Storage Lens データをモニタリングおよび分析することもできます。詳細については、次のトピックを参照してください。

- S3 Storage Lens のメトリクスとディメンション
- CloudWatch で S3 Storage Lens のメトリクスを表示します。

CloudWatch で S3 Storage Lens のメトリクスを表示します。

S3 ストレージレンズのメトリクスを Amazon CloudWatch に公開すると、CloudWatch ダッシュボードで運用状況を一元的に表示できます。また、アラームやトリガーアクション、メトリクス計算、異常検出などの CloudWatch 機能を使用して、S3 ストレージレンズのメトリクスをモニタリ

ングしてアクションを実行できます。さらに CloudWatch API オペレーションにより、サードパーティープロバイダーを含むアプリケーションが S3 ストレージレンズのメトリクスにアクセスできるようになります。CloudWatch Events の詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

Amazon S3 コンソール、Amazon S3 REST API、AWS CLI および AWS SDK を使用して、新規または既存のダッシュボード構成で CloudWatch 公開オプションを有効にすることができます。CloudWatch の公開オプションは、S3 ストレージレンズの高度なメトリクスとレコメンデーションにアップグレードされたダッシュボードで利用可能です。S3 ストレージレンズの高度なメトリクスとレコメンデーションの料金については、「Amazon S3 料金表」を参照してください。追加の CloudWatch メトリクスの公開料金は適用されません。ただし、ダッシュボード、アラーム、API コールなどのその他の CloudWatch 料金が適用されます。詳細については、Amazon CloudWatch の料金を参照してください。

S3 Storage Lens のメトリクスは、S3 Storage Lens の設定を所有するアカウント内の CloudWatch に公開されます。高度なメトリクスとレコメンデーションで CloudWatch 公開オプションを有効にすると、CloudWatch の組織、アカウント、バケットレベルのメトリクスにアクセスできます。プレフィックスレベルのメトリクスは CloudWatch では使用できません。

> **Note**
>
> S3 Storage Lens メトリクスは毎日のメトリクスであり、CloudWatch に 1 日 1 回公開されます。CloudWatch で S3 Storage Lens メトリクスをクエリする場合、クエリの期間は 1 日 (86400 秒) にする必要があります。毎日の S3 Storage Lens メトリクスが Amazon S3 コンソールの S3 Storage Lens ダッシュボードに表示された後、これらの同じメトリクスが CloudWatch に表示されるには数時間かかることがあります。初めて S3 Storage Lens メトリクスの CloudWatch 公開オプションを有効にすると、メトリクスが CloudWatch に公開されるまで最大で 24 時間かかることがあります。
> 現在、S3 ストレージレンズのメトリクスは CloudWatch ストリームを介して使用することはできません。

CloudWatch での S3 Storage Lens メトリクスの操作方法の詳細については、以下のトピックを参照してください。

トピック

- CloudWatch ダッシュボードの使用
- アラームの設定、アクションのトリガー、異常検出の使用

- ディメンションを使用した指標のフィルタリング
- メトリクス数学による新しいメトリクスの計算
- グラフでの検索式の使用

CloudWatch ダッシュボードの使用

CloudWatch ダッシュボードを使用して、S3 Storage Lens メトリクスを他のアプリケーションメトリクスとともにモニタリングし、運用状態の統一ビューを作成できます。ダッシュボードは、CloudWatch コンソールのカスタマイズ可能なホームページで、 リソースを単一のビューでリソースをモニタリングするために使用できます。

CloudWatch には、特定のメトリクスまたはディメンションへのアクセスの制限をサポートしない幅広い権限コントロールがあります。CloudWatch へのアクセス権を持つアカウントまたは組織のユーザーは、CloudWatch サポートオプションが有効になっているすべての S3 Storage Lens 設定のメトリクスにアクセスできます。S3 Storage Lens では、特定のダッシュボードのアクセス許可を管理することはできません。CloudWatch のアクセス許可の詳細については Amazon CloudWatch ユーザーガイドの CloudWatch リソースに対するアクセス許可の管理を参照してください。

CloudWatch ダッシュボードの使用とアクセス許可の設定の詳細については、 Amazon CloudWatch ユーザーガイドの Amazon CloudWatch ダッシュボードの使用および CloudWatch ダッシュボードの共有を参照してください。

アラームの設定、アクションのトリガー、異常検出の使用

CloudWatch で S3 Storage Lens メトリクスを監視し、しきい値を超えたときにアクションを実行する CloudWatch アラームを設定できます。例えば、未完了のマルチパートアップロードバイトメトリクスが 3 日間連続で 1 GB を超えた場合に Amazon SNS 通知を送信するアラームを設定できます。

また、異常検出を有効にして S3 Storage Lens のメトリクスを継続的に分析し、通常のベースラインを決定後、異常を検出できます。メトリクスの想定値に基づいて異常検出アラームを作成できます。例えば、オブジェクトロックが有効になっているバイト数メトリクスに異常がないかをモニタリングすると、オブジェクトロック設定の不正な削除を検出できます。

詳細な説明と例については、「Amazon CloudWatch アラームを使用する」そして 「Amazon CloudWatch ユーザーガイド」の「グラフのメトリクスからアラームを作成する」を参照してください。

ディメンションを使用した指標のフィルタリング

ディメンションを使用して、CloudWatch コンソールで S3 Storage Lens メトリクスをフィルタリングできます。例えば、次のようにフィルタリングできます。configuration_id、aws_account_number、aws_region、bucket_name、など。

S3 Storage Lens は、アカウントごとに複数のダッシュボード構成をサポートします。つまり、異なる構成に同じバケットを含めることができます。これらのメトリクスが CloudWatch に公開されると、バケットは CloudWatch 内に重複するメトリクスを持つことになります。CloudWatch の特定の S3 Storage Lens 設定のメトリクスだけを表示するには、configuration_id ディメンションを使用できます。configuration_id でフィルタリングすると、特定した構成に関連付けられたメトリクスのみが表示されます。

設定 ID によるフィルタリングの詳細については、「Amazon CloudWatch ユーザーガイド」の「利用可能なメトリクスの検索」を参照してください。

メトリクス数学による新しいメトリクスの計算

メトリクス数学により複数の S3 Storage Lens メトリクスをクエリし、数式を使用して、これらのメトリクスに基づく新しい時系列を作成できます。例えば、オブジェクト数から暗号化されたオブジェクト数を引いて、暗号化されていないオブジェクトの新しいメトリクスを作成できます。また、StorageBytes を ObjectCount で割って平均オブジェクトサイズを、BytesDownloaded を StorageBytes で割って 1 日にアクセスしたバイト数を取得するためのメトリクスを作成することもできます。

詳細については、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch メトリクス数学の使用」を参照してください。

グラフでの検索式の使用

S3 ストレージレンズのメトリクスを使用すると、検索式を作成することができます。例えば、IncompleteMultipartUploadStorageBytes と呼ばれるすべてのメトリクスの検索式を作成して、SUM を式に追加することができます。この検索式を使用すると、ストレージのすべてのディメンションで未完了のマルチパートアップロードバイトの合計を 1 つのメトリクスで確認できます。

この例では、IncompleteMultipartUploadStorageBytes と呼ばれるすべてのメトリクスの検索式を作成するために使用する構文を示します。

```
SUM(SEARCH('{AWS/S3/Storage-
Lens,aws_account_number,aws_region,configuration_id,metrics_version,record_type,storage_class}
 MetricName="IncompleteMultipartUploadStorageBytes"', 'Average',86400))
```

この構文の詳細については、「Amazon CloudWatch ユーザーガイド」の「CloudWatch 検索式の構文」を参照してください。検索式を使用した CloudWatch グラフを作成するには、「Amazon CloudWatch ユーザーガイド」の「検索式を使用した CloudWatch グラフの作成」を参照してください。

Amazon S3 ストレージレンズメトリクスのユースケース

Amazon S3 ストレージレンズのダッシュボードを使用して、インサイトと傾向を可視化し、外れ値にフラグを付け、推奨事項を受け取ることができます。S3 ストレージレンズメトリクスは、主要なユースケースに合致するカテゴリに整理されます。これらのメトリクスを使用して、次のことができます。

- コスト最適化の機会の特定
- データ保護のベストプラクティスの適用
- アクセス管理のベストプラクティスの適用
- アプリケーションワークロードのパフォーマンスの向上

例えば、コスト最適化メトリクスを使用して、Amazon S3 のストレージコストを削減する機会を特定することができます。7 日以上経過したマルチパートアップロードのあるバケットや、最新ではないバージョンが蓄積されているバケットを特定できます。

同様に、データ保護メトリクスを使用することで、組織内のデータ保護のベストプラクティスに従っていないバケットを特定できます。例えば、デフォルトの暗号化に AWS Key Management Service キー (SSE-KMS) を使用しないバケットや、S3 バージョニングが有効になっていないバケットを特定できます。

S3 ストレージレンズのアクセス管理メトリクスでは、S3 オブジェクト所有権のバケット設定を特定でき、アクセスコントロールリスト (ACL) アクセス許可をバケットポリシーに移行して ACL を無効にできます。

S3 ストレージレンズの高度なメトリクスを有効にしている場合は、詳細なステータスコードメトリクスを使用してリクエストの成功または失敗の数を取得し、アクセスやパフォーマンスの問題のトラブルシューティングに使用できます。

高度なメトリクスを使用すると、追加のコスト最適化メトリクスやデータ保護メトリクスにアクセスして、S3 ストレージ全体のコストをさらに削減し、データ保護のベストプラクティスとの整合性を高める機会を特定することもできます。例えば、高度なコスト最適化メトリクスにはライフサイクル

ルール数が含まれます。このメトリクスを使用して、7 日以上経過した不完全なマルチパートアップロードを失効させるライフサイクルルールがないバケットを特定できます。高度なデータ保護メトリクスには、レプリケーションルール数が含まれます。

メトリクスのカテゴリの詳細については、「メトリクスのカテゴリ」を参照してください。S3 ストレージレンズメトリクスの完全なリストについては、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

トピック

- Amazon S3 ストレージレンズを使用したストレージコストの最適化
- S3 ストレージレンズによるデータ保護
- オブジェクト所有権設定を監査するための S3 の使用
- S3 ストレージレンズのメトリクスを使用してパフォーマンスを改善する

Amazon S3 ストレージレンズを使用したストレージコストの最適化

S3 ストレージレンズのコスト最適化メトリクスを使用すると、S3 ストレージの全体的なコストを削減できます。コスト最適化メトリクスは、Amazon S3 をコスト効率よく、ベストプラクティスに従って設定したことを確認するのに役立ちます。例えば、次のようなコスト最適化の機会を特定できます。

- 7 日以上経過した未完了のマルチパートアップロードを含むバケット
- 多数の旧バージョンが蓄積されているバケット
- 未完了のマルチパートアップロードを中止するためのライフサイクルルールが設定されていないバケット
- 旧バージョンのオブジェクトを期限切れにするライフサイクルルールがないバケット
- オブジェクトを別のストレージクラスに移行するためのライフサイクルルールがないバケット

その後、このデータを使用して、バケットにライフサイクルルールを追加できます。

次の例は、S3 ストレージレンズダッシュボードでコスト最適化メトリクスを使用してストレージコストを最適化する方法を示しています。

トピック

- 最大の S3 バケットを特定する

- コールド Amazon S3 バケットを検出する
- 未完了のマルチパートアップロードを特定する
- 保持する旧バージョンの数を減らす
- ライフサイクルルールがないバケットを特定し、ライフサイクルルール数を確認する

最大の S3 バケットを特定する

料金は S3 バケットでのオブジェクトの保存に対して発生します。請求される料金は、オブジェクトのサイズ、オブジェクトの保存期間、およびそれらのストレージクラスに応じて異なります。S3 ストレージレンズで、アカウント内のすべてのバケットを一元的に表示できます。組織が持つすべてのアカウントのバケットをすべて表示するには、AWS Organizations レベルの S3 ストレージレンズダッシュボードを設定できます。最大のバケットは、このダッシュボードのビューから特定できます。

ステップ 1: 最大のバケットを特定する

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。

   ダッシュボードが開くと、S3 ストレージレンズがメトリクスを収集した最新の日付を確認できます。ダッシュボードでは常に、メトリクスの集計が可能な、最新の日付で読み込が行われます。
4. 選択した日付範囲の [Total storage] (ストレージの合計) メトリクスで最大バケットのランキングを表示するには、[Top N overview for date] (日付の上位 N の概要) セクションまでスクロールダウンします。

   ソート順序を切り替えて、最も小さいバケットを表示できます。また、[Metric] (メトリクス) の選択を調整して、使用可能なメトリクスのいずれかでバケットをランク付けすることもできます。[Top N overview for date] (日付の上位 N の概要) セクションには、前日または前週からの変化率と、トレンドを視覚化するスパークラインも表示されます。この傾向は、無料メトリクスの場合は 14 日間、高度なメトリクスとレコメンデーションの場合は 30 日間の傾向です。

> **Note**
>
> S3 ストレージレンズの高度なメトリクスとレコメンデーションの場合、メトリクスは 15 か月間クエリで利用できます。詳細については、「メトリクスの選択」を参照してください。

5. バケットに関するより詳細な情報については、ページの上部までスクロールし、[Bucket] (バケット) タブを選択します。

   [Buckets] (バケット) タブでは、最近の増加率、平均的なオブジェクトサイズ、最大のプレフィックス、およびオブジェクトの数などの詳細を確認できます。

ステップ 2: バケットに移動して調査する

最大の S3 バケットを特定したら、S3 コンソール内の各バケットに移動してバケット内のオブジェクトを表示し、関連するワークロードを理解し、内部所有者を特定できます。バケットの所有者に連絡すれば、この増加が見込まれていたものかどうか、またはこの増加をさらにモニタリングして制御する必要があるかどうかを確認できます。

コールド Amazon S3 バケットを検出する

S3 ストレージレンズの高度なメトリクス を有効にしている場合は、アクティビティメトリクスを使用して、S3 バケットがどの程度「コールド」かを理解できます。「コールド」なバケットとは、そのストレージがアクセスされなくなった (またはほとんどアクセスされない) バケットです。通常、このアクティビティの欠如はバケットのオブジェクトが頻繁にアクセスされていないことを示します。

[GET Requests] (GET リクエスト) および [Download Bytes] (ダウンロード済みバイト数) などのアクティビティメトリクスは、バケットが毎日どのくらいの頻度でアクセスされるかを示します。アクセスパターンの一貫性を理解し、まったくアクセスされなくなったバケットを特定するために、このデータの数か月間にわたる傾向を得ることができます。[Download bytes / Total storage] (ダウンロード済みバイト数÷ストレージの合計) として計算される [Retrieval rate] (取得率) メトリクスは、毎日アクセスされるバケット内のストレージの比率を示します。

> **Note**
>
> 1 日に同じオブジェクトが複数回ダウンロードされる場合は、ダウンロードのバイト数が重複します。

前提条件

S3 ストレージレンズダッシュボードでアクティビティメトリクスを表示するには、S3 ストレージレンズ [Advanced metrics and recommendations] (高度なメトリクスとレコメンデーション) を有効にしてから、[Activity metrics] (アクティビティメトリクス) を選択する必要があります。詳細については、「Amazon S3 Storage Lens ダッシュボードの作成と更新」を参照してください。

ステップ 1: アクティブなバケットを特定する

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. [Bucket] (バケット) タブを選択してから、[Bubble analysis by buckets for date] (日付別のバケット別によるバブル分析) セクションまでスクロールダウンします。

   [Bubble analysis by buckets for date] (日付別のバケット別によるバブル分析) セクションでは、バブルの [X-axis] (X 軸)、[Y-axis] (Y 軸)、および [Size] (サイズ) を表す 3 つのメトリクスを使用して、バケットを複数のディメンションにプロットできます。
5. コールドになったバケットを見つけるには、[X-axis] (X 軸)、[Y-axis] (Y 軸)、および [Size] (サイズ) で、[Total storage] (合計ストレージ)、[% retrieval rate] (% 取り出し率)、および [Average object size] (平均オブジェクトサイズ) のメトリクスを選択します。
6. [Bubble analysis by buckets for date] (日付別のバケットごとのバブル分析) セクションで、取得率がゼロ (またはゼロに近い) で、かつ相対的なストレージサイズが大きいバケットを探し、そのバケットを表すバブルを選択します。

   より詳細なインサイトの選択肢が記載されたボックスが表示されます。以下のいずれかを実行します。

   a. [Bucket] (バケット) タブを更新して、選択したバケットのメトリクスのみを表示するには、[Drill down] (ドリルダウン)、[Apply] (適用) を順に選択します。
   b. バケットレベルのデータをアカウント、AWS リージョン、ストレージクラス、またはバケットごとに集約するには、[Analyze by] (分析基準)、[Dimension] (ディメンション) の順に選択します。例えば、ストレージクラス別に集計するには、[Dimension] (ディメンション) に [Storage class] (ストレージクラス) を選択します。

コールドになったバケットを見つけるには、[Total storage] (ストレージの合計)、[% retrieval rate] (% 取得率)、および [Average object size] (平均オブジェクトサイズ) メトリクスを使用してバブル分析を実行します。取得率がゼロ (またはゼロに近い) で、かつ相対的なストレージサイズが大きいバケットを探します。

ダッシュボードの [Bucket] (バケット) タブが更新され、選択した集計またはフィルターのデータが表示されます。ストレージクラスまたは別のディメンション別に集計した場合、その新しいタブがダッシュボードに表示されます (例えば、[Storage class] (ストレージクラス) タブ)。

ステップ 2: コールドバケットを調査する

そこからアカウントまたは組織内のコールドバケットの所有者を特定し、そのストレージが引き続き必要かどうかを確認できます。その後、バケットにライフサイクル有効期限設定を設定する、またはデータを Amazon S3 Glacier ストレージクラスにアーカイブすることによって、コストを最適化できます。

将来のコールドバケット問題を回避するには、S3 ライフサイクル設定を使用してデータを自動的に移行させる、または S3 Intelligent-Tiering による自動アーカイブを有効にすることができます。

ステップ 1 を実施してホットバケットを識別することもできます。次に、これらのバケットが正しい S3 ストレージクラスを使用するようにして、パフォーマンスとコストの面で最も効果的にリクエストに応えられるようにします。

未完了のマルチパートアップロードを特定する

スループットを向上させ、ネットワーク問題からより迅速に復旧するため、マルチパートアップロードを使用して超大型のオブジェクト (最大 5 TB) を複数のパートから成るセットとしてアップロードすることができます。マルチパートアップロードのプロセスが完了しなかった場合、未完了のなパートが使用不可能な状態でバケットに残ります。これらの未完了のパートには、アップロードプロセスが完了するか、未完了の部分が削除されるまで、ストレージコストが発生します。詳細については、「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。

S3 ストレージレンズを使用すると、7 日以上経過した未完了のマルチパートアップロードを含め、アカウント内、または組織全体にある未完了のマルチパートアップロードのバイト数を特定できます。未完了のマルチパートアップロードメトリクスの完全なリストについては、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

ベストプラクティスとして、特定の日数を過ぎた未完了のマルチパートアップロードを期限切れにするように、ライフサイクルルールを設定することをおすすめします。未完了のマルチパートアップロードを期限切れにするライフサイクルルールを作成する場合は、7 日間から始めることをお勧めします。

ステップ 1: 未完了のマルチパートアップロードの全体的なトレンドを確認する

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. [Snapshot for date] (日付のスナップショット) セクションの [Metrics categories] (メトリクスのカテゴリ) で、[Cost optimization] (コスト最適化) を選択します。

   [Snapshot for date] (日付のスナップショット) セクションが更新され、[Incomplete multipart upload bytes greater than 7 days old] (7 日以上経過した未完了のマルチパートアップロードバイト数) を含む [Cost optimization] (コスト最適化) メトリクスが表示されます。

   S3 ストレージレンズダッシュボードのどのグラフでも、未完了のマルチパートアップロードのメトリクスを確認できます。これらのメトリクスを使用して、全体的な増加傾向に対するそれらの寄与度などを含め、ストレージに対する未完了のマルチパートアップロードのバイト数の影響をさらに評価することができます。また、[Account] (アカウント)、[AWS リージョン]、[Bucket] (バケット)、[Storage class] (ストレージクラス) の各タブを使用して、より詳細な集計レベルまで掘り下げてデータを分析することもできます。例については、「コールド Amazon S3 バケットを検出する」を参照してください。

ステップ 2: マルチパートアップロードバイト数が最も多く、未完了のマルチパートアップロードを中止するライフサイクルルールがないバケットを特定する

前提条件

S3 ストレージレンズダッシュボードに [Abort incomplete multipart upload lifecycle rule count] (未完了のマルチパートアップロードの中止ライフサイクルルール数) メトリクスを表示するには、S3 ストレージレンズの [Advanced metrics and recommendations] (高度なメトリクスとレコメンデーション) を有効にしてから [Advanced cost optimization metrics] (高度なコスト最適化メトリクス) を選択する必要があります。詳細については、「Amazon S3 Storage Lens ダッシュボードの作成と更新」を参照してください。

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. 7 日以上経過した未完了のマルチパートアップロードが蓄積されているバケットを特定するには、[Top N overview for date] (日付の上位 N の概要) セクションに移動します。

   デフォルトでは、[Top N overview for date] (日付の上位 N の概要) セクションには、上位 3 つのバケットのメトリクスが表示されます。[Top N] (トップ N) のフィールドでバケット数を増減できます。[Top N overview for date] (日付の上位 N の概要) セクションには、前日または前週からの変化率と、トレンドを視覚化するスパークラインも表示されます。(このトレンドは、無料メトリクスの場合は 14 日間、高度なメトリクスとレコメンデーションの場合は 30 日間のトレンドです。)

   > **Note**
   >
   > S3 ストレージレンズの高度なメトリクスとレコメンデーションの場合、メトリクスは 15 か月間クエリで利用できます。詳細については、「メトリクスの選択」を参照してください。

5. [Metric] (メトリクス) で、[Cost optimization] (コスト最適化) カテゴリの [Incomplete multipart upload bytes greater than 7 days old] (7 日以上経過した未完了のマルチパートアップロードバイト数) を選択します。

   [Top number buckets] (上位バケット) には、7 日以上経過している未完了のマルチパートアップロードのストレージバイト数が最も多いバケットが表示されます。
6. 未完了のマルチパートアップロードに関する詳細なバケットレベルのメトリクスを表示するには、ページの一番上までスクロールして、[Bucket] (バケット) タブを選択します。
7. [Buckets] (バケット) セクションまで下にスクロールします。[Metrics categories] (メトリクスのカテゴリ) で、[Cost optimization] (コスト最適化) を選択します。次に , [Summary] (概要) を選択解除します。

   [Buckets] (バケット) リストが更新され、表示されているバケットで利用可能なすべての [Cost optimization] (コスト最適化) メトリクスが表示されます。
8. [Buckets] (バケット) リストをフィルタリングして特定のコスト最適化メトリクスのみを表示するには、設定アイコン

( )
を選択します。

9. [Incomplete multipart upload bytes greater than 7 days old] (7 日以上経過した未完了のマルチパートアップロードバイト数) および [Abort incomplete multipart upload lifecycle rule count] (未完了のマルチパートアップロードの中止ライフサイクルルール数) 以外のすべてのコスト最適化メトリクスのトグルを選択解除します。
10. (オプション) [Page size] (ページサイズ) で、リストに表示するバケットの数を選択します。
11. [確認] を選択します。

    [Buckets] (バケット) リストが更新され、未完了のマルチパートアップロードのバケットレベルのメトリクスとライフサイクルルール数が表示されます。このデータを使用して、7 日以上経過した未完了のマルチパートアップロードバイト数が最も多く、未完了のマルチパートアップロードを中止するライフサイクルルールが欠落しているバケットを特定できます。次に、S3 コンソールでこれらのバケットに移動し、ライフサイクルルールを追加して、放棄された未完了のマルチパートアップロードを削除できます。

ステップ 3: 未完了のマルチパートアップロードを 7 日後に削除するライフサイクルルールを追加する

未完了のマルチパートアップロードを自動的に管理するには、S3 コンソールを使用して、指定された日数後にバケットから未完了のマルチパートアップロードのバイトを失効させるライフサイクル設定を作成できます。詳細については、「不完全なマルチパートアップロードを削除するためのバケットライフサイクル設定の設定」を参照してください。

保持する旧バージョンの数を減らす

S3 バージョニング機能を有効にすると、同じオブジェクトの複数の異なるバージョンが保持されます。これらは、オブジェクトが誤って削除または上書きされた場合にデータをすばやく回復させるために使用できます。旧バージョンを移行または期限切れにするライフサイクルルールを設定せずに S3 バージョニングを有効にした場合、旧バージョンが大量に蓄積され、ストレージコストに影響する可能性があります。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

ステップ 1: 最新でないオブジェクトバージョンが最も多いバケットを特定する

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。

2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. [Snapshot for date] (日付のスナップショット) セクションの [Metrics categories] (メトリクスのカテゴリ) で、[Cost optimization] (コスト最適化) を選択します。

   [Snapshot for date] (日付のスナップショット) セクションが更新され、[Cost optimization] (コスト最適化) メトリクスが表示されます。これには、[% noncurrent version bytes] (旧バージョンのバイト数 %) メトリクスが含まれます。[% noncurrent version bytes] (旧バージョンのバイト数 %) のメトリクスは、ダッシュボードの範囲内かつ選択された日付に関して、ストレージの合計バイト数のうち、旧バージョンに起因するバイト数の割合を表します。

   > **Note**
   >
   > [% noncurrent version bytes] (旧バージョンのバイト数 %) がアカウントレベルでストレージの 10 パーセントを超過する場合、保存しているオブジェクトバージョンの数が多すぎる可能性があります。

5. 旧バージョンが大量に蓄積されているバケットを特定するには:

   a. [Top N overview for date] (日付の上位 N の概要) セクションまで下にスクロールします。[Top N] (上位 N) には、データを表示したいバケットの数を入力します。

   b. [Metric] (メトリクス) で、[% noncurrent version bytes] (旧バージョンのバイト数 %) を選択します。

      [Top number buckets] (バケットの上位) の下に、[% noncurrent version bytes] (旧バージョンのバイト数 %) が最も高い (指定した数の) バケットが表示されます。[Top N overview for date] (日付の上位 N の概要) セクションには、前日または前週からの変化率と、トレンドを視覚化するスパークラインも表示されます。この傾向は、無料メトリクスの場合は 14 日間、高度なメトリクスとレコメンデーションの場合は 30 日間の傾向です。

      > **Note**
      >
      > S3 ストレージレンズの高度なメトリクスとレコメンデーションの場合、メトリクスは 15 か月間クエリで利用できます。詳細については、「メトリクスの選択」を参照してください。

c. 最新でないオブジェクトバージョンに関する詳細なバケットレベルのメトリクスを表示するには、ページの一番上までスクロールして、[Bucket] (バケット) タブを選択します。

S3 ストレージレンズダッシュボードのどのグラフや可視化でも、[Account] (アカウント)、[AWS リージョン]、[Storage class] (ストレージクラス)、または [Bucket] (バケット) タブを使用して、より深いレベルの集計にドリルダウンできます。例については、「コールド Amazon S3 バケットを検出する」を参照してください。

d. [Buckets] (バケット) セクションの [Metric categories] (メトリクスカテゴリ) で、[Cost optimization] (コスト最適化) を選択します。次に。[Summary] (概要) を選択解除します。

旧バージョンに関連する他のメトリクスと共に、[% noncurrent version bytes] (旧バージョンのバイト数 %) メトリクスを確認できるようになりました。

ステップ 2: 旧バージョンを管理するための移行および有効期限のライフサイクルルールがないバケットを特定する

前提条件

S3 ストレージレンズダッシュボードで [Noncurrent version transition lifecycle rule count] (旧バージョン移行ライフサイクルルール数) および [Noncurrent version expiration lifecycle rule count] (旧バージョン有効期限ライフサイクルルール数) メトリクスを確認するには、S3 ストレージレンズの [Advanced metrics and recommendations] (高度なメトリクスとレコメンデーション)を有効にしてから、[Advanced cost optimization metrics] (高度なコスト最適化メトリクス) を選択する必要があります。詳細については、「Amazon S3 Storage Lens ダッシュボードの作成と更新」を参照してください。

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. ストレージレンズダッシュボードで、[Bucket] (バケット) タブを選択します。
5. [Buckets] (バケット) セクションまで下にスクロールします。[Metrics categories] (メトリクスのカテゴリ) で、[Cost optimization] (コスト最適化) を選択します。次に , [Summary] (概要) を選択解除します。

[Buckets] (バケット) リストが更新され、表示されているバケットで利用可能なすべての [Cost optimization] (コスト最適化) メトリクスが表示されます。

6. [Buckets] (バケット) リストをフィルタリングして特定のコスト最適化メトリクスのみを表示するには、設定アイコン
()
を選択します。
7. 次の項目だけが選択された状態になるまで、他のすべてのコスト最適化メトリクスを選択解除します。

   - % noncurrent version bytes (旧バージョンのバイト数 %)
   - Noncurrent version transition lifecycle rule count (旧バージョン移行ライフサイクルルール数)
   - Noncurrent version expiration lifecycle rule count (旧バージョン有効期限ライフサイクルルール数)
8. (オプション) [Page size] (ページサイズ) で、リストに表示するバケットの数を選択します。
9. [確認] を選択します。

   [Buckets] (バケット)リストが更新され、旧バージョンのバイト数と旧バージョンのライフサイクルルール数のメトリクスが表示されます。このデータを使用して、旧バージョンのバイト数の割合が高いが、移行および有効期限のライフサイクルルールが欠落しているバケットを特定できます。次に、S3 コンソールでこれらのバケットに移動し、これらのバケットにライフサイクルルールを追加できます。

ステップ 3: ライフサイクルルールを追加して最新でないオブジェクトバージョンを移行または期限切れにする

追加の調査が必要なバケットを判断したら、S3 コンソールでそのバケットに移動し、指定した日数後に旧バージョンを失効させるライフサイクルルールを追加できます。あるいは、旧バージョンを保持したままコストを削減するために、旧バージョンを Amazon S3 Glacier ストレージクラスのいずれかに移行するライフサイクルルールを設定することもできます。詳細については、「例 6: バージョニングが有効なバケットへのライフサイクルルールを指定する」を参照してください。

ライフサイクルルールがないバケットを特定し、ライフサイクルルール数を確認する

S3 ストレージレンズには、ライフサイクルルールがないバケットを特定するために使用できる S3 ライフサイクルルール数メトリクスが用意されています。ライフサイクルルールがないバケットを見つけるには、[Total buckets without lifecycle rules] (ライフサイクルルールがないバケットの合計数)

メトリクスを使用できます。S3 ライフサイクル設定のないバケットには、不要になったストレージや、低コストのストレージクラスに移行できるストレージがある可能性があります。ライフサイクルルール数メトリクスを使用して、有効期限ルールや移行ルールなど、特定の種類のライフサイクルルールが欠落しているバケットを特定することもできます。

前提条件

S3 ライフサイクルダッシュボードでライフサイクルルール数のメトリクスと [Total buckets without lifecycle rules] (ライフサイクルルールがないバケットの合計数) メトリクスを表示するには、S3 ストレージレンズの [Advanced metrics and recommendations] (高度なメトリクスとレコメンデーション) を有効にしてから [Advanced cost optimization metrics] (高度なコスト最適化メトリクス) を選択する必要があります。詳細については、「Amazon S3 Storage Lens ダッシュボードの作成と更新」を参照してください。

ステップ 1: ライフサイクルルールが設定されていないバケットを特定する

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. ライフサイクルルールのない特定のバケットを確認するには、[Top N overview for date] (日付の上位 N の概要) セクションまで下にスクロールします。

   デフォルトでは、[Top N overview for date] (日付の上位 N の概要) セクションには、上位 3 つのバケットのメトリクスが表示されます。[Top N] (トップ N) のフィールドでバケット数を増やすことができます。[Top N overview for date] (日付のトップ N の概要) セクションには、前日または前週からの変化率と、傾向を視覚化するスパークラインも表示されます。この傾向は、無料メトリクスの場合は 14 日間、高度なメトリクスとレコメンデーションの場合は 30 日間の傾向です。

   > **Note**
   >
   > S3 ストレージレンズの高度なメトリクスとレコメンデーションの場合、メトリクスは 15 か月間クエリで利用できます。詳細については、「メトリクスの選択」を参照してください。

5. [Metric] (メトリクス) で、[Total buckets without lifecycle rules] (ライフサイクルルールがないバケットの合計数) を [Cost optimization] (コスト最適化) カテゴリから選択します。
6. [Total buckets without lifecycle rules] (ライフサイクルルールがないバケットの合計数) については、次のデータを確認してください:

   - Top number accounts (アカウント上位) – ライフサイクルルールのないバケット数が最も多いアカウントを確認できます。
   - Top number Regions (地域の上位) – ライフサイクルルールのないバケットの内訳を地域別に表示します。
   - Top number buckets (バケットの上位) – どのバケットにライフサイクルルールがないかを確認できます。

   S3 ストレージレンズダッシュボードのどのグラフや可視化でも、[Account] (アカウント)、[AWS リージョン]、[Storage class] (ストレージクラス)、または [Bucket] (バケット) タブを使用して、より深いレベルの集計にドリルダウンできます。例については、「コールド Amazon S3 バケットを検出する」を参照してください。

   ライフサイクルルールがないバケットを特定したら、バケットの特定のライフサイクルルール数を確認することもできます。

ステップ 2: バケットのライフサイクルルール数を確認する

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. S3 ストレージレンズダッシュボードで、[Bucket] (バケット) タブを選択します。
5. [Buckets] (バケット) セクションまで下にスクロールします。[Metrics categories] (メトリクスのカテゴリ) で、[Cost optimization] (コスト最適化) を選択します。次に , [Summary] (概要) を選択解除します。

   [Buckets] (バケット) リストが更新され、表示されているバケットで利用可能なすべての [Cost optimization] (コスト最適化) メトリクスが表示されます。
6. [Buckets] (バケット) リストをフィルタリングして特定のコスト最適化メトリクスのみを表示するには、設定アイコン

()
を選択します。

7. 次の項目だけが選択された状態になるまで、他のすべてのコスト最適化メトリクスを選択解除します。

   - Transition lifecycle rule count (移行ライフサイクルルール数)
   - Expiration lifecycle rule count (有効期限ライフサイクルルール数)
   - Noncurrent version transition lifecycle rule count (旧バージョン移行ライフサイクルルール数)
   - Noncurrent version expiration lifecycle rule count (旧バージョン有効期限ライフサイクルルール数)
   - Abort incomplete multipart upload lifecycle rule count (未完了のマルチパートアップロードの中止ライフサイクルルール数)
   - Total lifecycle rule count (ライフサイクルルールの合計数)
8. (オプション) [Page size] (ページサイズ) で、リストに表示するバケットの数を選択します。
9. [確認] を選択します。

   [Buckets] (バケット) リストが更新され、バケットのライフサイクルルール数メトリクスが表示されます。このデータを使用して、ライフサイクルルールがないバケットや、有効期限や移行ルールなどの特定の種類のライフサイクルルールがないバケットを特定できます。次に、S3 コンソールでこれらのバケットに移動し、これらのバケットにライフサイクルルールを追加できます。

ステップ 3: ライフサイクルルールを追加する

ライフサイクルルールのないバケットを特定したら、ライフサイクルルールを追加できます。詳細については、バケットにライフサイクル設定を設定する および S3 ライフサイクル設定の例 を参照してください。

S3 ストレージレンズによるデータ保護

Amazon S3 ストレージレンズのデータ保護メトリクスを使用して、データ保護のベストプラクティスが適用されていないバケットを特定できます。これらのメトリクスを使用して対策を講じ、ベストプラクティスに沿った標準設定を適用して、アカウントや組織内のさまざまなバケットにわたってデータを保護できます。例えば、データ保護メトリクスを使用して、デフォルトの暗号化に AWS Key Management Service (AWS KMS) キー (SSE-KMS) を使用しないバケットや、AWS Signature Version 2 (SigV2) を使用するリクエストを特定できます。

以下のユースケースでは、S3 ストレージレンズダッシュボードを使用して外れ値を特定し、S3 バケット全体にデータ保護のベストプラクティスを適用する戦略を提供します。

トピック

- AWS KMS によるデフォルトの暗号化 (SSE-KMS) でサーバー側暗号化を使用しないバケットを識別する
- S3 バージョニングが有効なバケットを識別する
- AWS Signature Version 2 (SigV2) を使用したリクエストを識別する
- 各バケットのレプリケーションルールの総数を数える
- オブジェクトロックバイトのパーセンテージを識別する

AWS KMS によるデフォルトの暗号化 (SSE-KMS) でサーバー側暗号化を使用しないバケットを識別する

Amazon S3 のデフォルトの暗号化を使用して、S3 バケットのデフォルト暗号化の動作を設定できます。詳細については、「the section called “デフォルトのバケット暗号化の設定”」を参照してください。

[SSE-KMS enabled bucket count] (SSE-KMS 対応バケット数) と[% SSE-KMS enabled buckets] (% SSE-KMS 対応バケット) メトリクスを使用して、デフォルトの暗号化に AWS KMS キー (SSE-KMS) によるサーバー側暗号化を使用しているバケットを特定できます。S3 ストレージレンズは、暗号化されていないバイト、暗号化されていないオブジェクト、暗号化されたバイト、および暗号化されたオブジェクトのメトリクスも提供します。メトリクスの一覧については、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

SSE-KMS 暗号化メトリクスを一般的な暗号化メトリクスのコンテキストで分析すると、SSE-KMS を使用していないバケットを特定できます。アカウントまたは組織のすべてのバケットに SSE-KMS を使用する場合は、これらのバケットのデフォルトの暗号化設定を SSE-KMS を使用するように更新できます。SSE-KMS に加え、Amazon S3 で管理されたキー (SSE-S3) またはお客様から提供されたキー (SSE-C) によりサーバー側の暗号化を使用できます。詳細については、「暗号化によるデータの保護」を参照してください。

ステップ 1: どのバケットがデフォルトの暗号化に SSE-KMS を使用しているかを特定する

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。

2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. [Trends and distributions] (傾向とディストリビューション) セクションで、プライマリメトリクスには [% SSE-KMS enabled bucket count] (% SSE-KMS 対応バケット数) を選択し、セカンダリメトリクスには [% encrypted bytes] (% 暗号化バイト数) を選択します。

   [Trend for date] (日付の傾向) グラフが更新され、SSE-KMS と暗号化されたバイトの傾向が表示されます。
5. SSE-KMS の詳細なバケットレベルのインサイトを表示するには:

   a. グラフ上のポイントを選択します。より詳細なインサイトの選択肢が記載されたボックスが表示されます。
   b. [Buckets] (バケット) ディメンションを選択します。次に、[Apply] (適用) を選択します。
6. [Distribution by buckets for date] (日付のバケット別分布) グラフで、[SSE-KMS enabled bucket count] (SSE-KMS 対応のバケット数) メトリクスを選択します。
7. これで、SSE-KMS が有効になっているバケットと有効になっていないバケットを確認できます。

ステップ 2: バケットのデフォルトの暗号化設定を更新する

どのバケットが [% encrypted bytes] (% 暗号化バイト) のコンテキストで SSE-KMS を使用するかを確認できたので、SSE-KMS を使用しないバケットを特定できます。その後、オプションで S3 コンソール内のこれらのバケットに移動し、デフォルトの暗号化設定を SSE-KMS または SSE-S3 を使用するように更新できます。詳細については、「デフォルトの暗号化の設定」を参照してください。

S3 バージョニングが有効なバケットを識別する

S3 バージョニング機能を有効にすると、同じオブジェクトの複数のバージョンが保持されます。これらは、オブジェクトが誤って削除された、または上書きされた場合にデータをすばやく回復させるために使用できます。[Versioning-enabled bucket count] (バージョニングが有効なバケット数) メトリクスを使用して、どのバケットが S3 バージョニングを使用しているかを確認できます。次に、S3 コンソールでアクションを実行して、他のバケットの S3 バージョニングを有効にできます。

ステップ 1: S3 バージョニングが有効なバケットを識別する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

2. ナビゲーションペインで、[Storage Lens]、[ダッシュボード] の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. [Trends and distributions] (傾向とディストリビューション) セクションで、プライマリメトリクスには [Versioning-enabled bucket count] (バージョニングが有効なバケット数) を選択し、セカンダリメトリクスには [Buckets] (バケット) を選択します。

   [Trend for date] (日付の傾向) グラフが更新され、S3 バージョニングが有効なバケットの傾向が表示されます。トレンドラインのすぐ下に、[Storage class distribution] (ストレージクラスディストリビューション) と[Region distribution] (リージョンディストリビューション) サブセクションが表示されます。
5. [Trend for date] (日付の傾向) グラフに表示されているバケットについてより詳細なインサイトを表示して、より詳細な分析を実行できるようにするには、次の操作を行います。

   a. グラフ上のポイントを選択します。より詳細なインサイトの選択肢が記載されたボックスが表示されます。

   b. データに適用するディメンションを [Account] (アカウント)、[AWS リージョン]、[Storage class] (ストレージクラス)、[Bucket] (バケット) から選択して、より詳細な分析を行います。次に、[Apply] (適用) を選択します。
6. [Bubble analysis by buckets for date] (日付別のバケットによるバブル分析) セクションで、[Versioning-enabled bucket count] (バージョニングが有効なバケット数)、[Buckets] (バケット)、[Active buckets] (アクティブバケット) のメトリクスを選択します。

   [Bubble analysis by buckets for date] (日付別のバケットごとのバブル分析) セクションが更新され、選択したメトリクスのデータが表示されます。このデータを使用して、合計バケット数に関連して、どのバケットで S3 バージョニングが有効になっているかを確認できます。[Bubble analysis by buckets for date] (日付のバケットによるバブル分析) セクションでは、バブルの [X-axis](X 軸)、[Y-axis] (Y 軸)、および [Size] (サイズ) を表す 3 つのメトリクスを使用して、バケットを複数のディメンションにプロットできます。

ステップ 2: S3 バージョニングを有効にする

S3 バージョニングが有効になっているバケットを特定したら、S3 バージョニングが有効になっていないバケット、またはバージョニングが停止されているバケットを特定できます。次に、オプションで S3 コンソールでこれらのバケットのバージョニングを有効にできます。詳細については、「バケットでのバージョニングの有効化」を参照してください。

AWS Signature Version 2 (SigV2) を使用したリクエストを識別する

[All unsupported signature requests] (すべてのサポートされていない署名リクエスト) メトリクスを使用して、AWS Signature Version 2 (SigV2) を使用するリクエストを特定できます。このデータは、SigV2 を使用している特定のアプリケーションを特定するのに役立ちます。その後、これらのアプリケーションを AWS Signature Version 4 (SigV4) に移行できます。

SigV4 はすべての新しい S3 アプリケーションに推奨される署名方法です。SigV4 はセキュリティが向上し、すべての AWS リージョン でサポートされています。詳細については、「Amazon S3 更新 - SigV2 の非推奨期間の延長および変更」を参照してください。

前提条件

[All unsupported signature requests] (すべてのサポートされていない署名リクエスト) を S3 ストレージレンズダッシュボードに表示するには、S3 ストレージレンズの [Advanced metrics and recommendations] (高度なメトリクスとレコメンデーション) を有効にしてから、[Advanced data protection metrics] (高度なデータ保護メトリクス) を選択します。詳細については、「Amazon S3 Storage Lens ダッシュボードの作成と更新」を参照してください。

ステップ 1: SigV2 署名の傾向を AWS アカウント、リージョン、バケット別に調べる

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. SigV2 を使用するリクエストがある特定のバケット、アカウント、リージョンを特定するには:

    a. [Top N] (トップ N) の [Top N overview for date] (日付のトップ N 概要) に、データを表示したいバケットの数を入力します。

    b. [Metric] (メトリクス) では、[Data protection] (データ保護) カテゴリから [All unsupported signature requests] (すべてのサポートされていない署名リクエスト) を選択します。

    [Top N overview for date] (日付のトップ N 概要) が更新され、アカウント、AWS リージョン、バケット別に SigV2 リクエストのデータが表示されます。[Top N overview for date] (日付のトップ N の概要) セクションには、前日または前週からの変化率と、傾向を視覚化するスパークラインも表示されます。この傾向は、無料メトリクスの場合は 14 日間、高度なメトリクスとレコメンデーションの場合は 30 日間の傾向です。

> **Note**
>
> S3 ストレージレンズの高度なメトリクスとレコメンデーションの場合、メトリクスは 15 か月間クエリで利用できます。詳細については、「メトリクスの選択」を参照してください。

ステップ 2: SigV2 リクエストを通じてアプリケーションからアクセスされるバケットを特定する

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. ストレージレンズダッシュボードで、[Bucket] (バケット) タブを選択します。
5. [Buckets] (バケット) セクションまで下にスクロールします。[Metrics categories] (メトリクスのカテゴリ) で、[Data protection] (データ保護) を選択します。次に、[Summary] (概要) を選択解除します。

   [Buckets] (バケット) リストが更新され、表示されているバケットで利用可能なすべての [Data protection] (データ保護) メトリクスが表示されます。
6. [Buckets] (バケット) リストをフィルタリングして特定のデータ保護メトリクスのみを表示するには、設定アイコン
   (
)
   を選択します。
7. 次の項目だけが選択された状態になるまで、他のすべてのデータ保護メトリクスを選択解除します。

   - [All unsupported signature requests] (すべてのサポートされていない署名リクエスト)
   - [% all unsupported signature requests] (% すべてのサポートされていない署名リクエスト)
8. (オプション) [Page size] (ページサイズ) で、リストに表示するバケットの数を選択します。
9. [確認] を選択します。

   [Buckets] (バケット) リストが更新され、SigV2 リクエストのバケットレベルのメトリクスが表示されます。このデータを使用して、SigV2 リクエストがある特定のバケットを特定でき

ます。次に、この情報を使用してアプリケーションを SigV4 に移行できます。詳細については、Amazon Simple Storage Service API リファレンスの「リクエストの認証 (AWS 署名バージョン 4)」を参照してください。

各バケットのレプリケーションルールの総数を数える

S3 レプリケーションを使用すると、Amazon S3 バケット間でオブジェクトを自動で非同期的にコピーできます。オブジェクトのレプリケーション用に設定されたバケットは、同じ AWS アカウントが所有することも、異なるアカウントが所有することもできます。詳細については、「オブジェクトのレプリケーション」を参照してください。

S3 ストレージレンズのレプリケーションルール数メトリクスを使用して、レプリケーション用に設定されているバケットに関する詳細なバケットごとの情報を取得できます。この情報には、バケット内とリージョン内およびバケット間とリージョン間のレプリケーションルールが含まれます。

前提条件

レプリケーションルール数メトリクスを S3 ストレージレンズダッシュボードに表示するには、S3 ストレージレンズ [Advanced metrics and recommendations] (高度なメトリクスとレコメンデーション) を有効にしてから、[Advanced data protection metrics] (高度なデータ保護メトリクス) を選択します。詳細については、「Amazon S3 Storage Lens ダッシュボードの作成と更新」を参照してください。

ステップ 1: 各バケットのレプリケーションルールの総数を数える

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. ストレージレンズダッシュボードで、[Bucket] (バケット) タブを選択します。
5. [Buckets] (バケット) セクションまで下にスクロールします。[Metrics categories] (メトリクスのカテゴリ) で、[Data protection] (データ保護) を選択します。次に、[Summary] (概要) を選択解除します。
6. [Buckets] (バケット) リストをフィルタリングしてレプリケーションルール数メトリクスのみを表示するには、設定アイコン

()
を選択します。

7. レプリケーションルール数メトリクスだけが選択された状態になるまで、他のすべてのデータ保護メトリクスを選択解除します。
   - [Same-Region Replication rule count] (同一リージョンレプリケーションのルール数)
   - [Cross-Region Replication rule count] (クロスリージョンレプリケーションルール数)
   - [Same-account replication rule count] (同一アカウントレプリケーションのルール数)
   - [Cross-account replication rule count] (クロスリージョンレプリケーションルール数)
   - [Total replication rule count] (レプリケーションルールの合計数)
8. (オプション) [Page size] (ページサイズ) で、リストに表示するバケットの数を選択します。
9. [確認] を選択します。

ステップ 2: レプリケーションルールを追加する

バケットごとのレプリケーションルール数が決まったら、オプションで追加のレプリケーションルールを作成できます。詳細については、「ライブレプリケーションの設定例」を参照してください。

オブジェクトロックバイトのパーセンテージを識別する

S3 オブジェクトロックでは、write-once-read-many (WORM) モデルを使用してオブジェクトを保存できます。オブジェクトロックを使用して、オブジェクトが削除または上書きされることを、一定期間または無期限に防止できます。オブジェクトロックは、バケットを作成する場合にのみ有効にでき、S3 バージョニングも有効化できます。ただし、個々のオブジェクトバージョンの保存期間を編集したり、オブジェクトロックが有効になっているバケットにリーガルホールドを適用したりできます。詳細については、「S3 オブジェクトロックの使用」を参照してください。

S3 ストレージレンズのオブジェクトロックメトリクスを使用して、アカウントまたは組織の [% Object Lock bytes] (% オブジェクトロックバイト) メトリクスを確認できます。この情報を使用して、アカウントまたは組織内のどのバケットがデータ保護のベストプラクティスに従っていないかを特定できます。

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。

3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. [Snapshot] (スナップショット) セクションの [Metrics categories] (メトリクスのカテゴリ) で、[Data protection] (データ保護) を選択します。

   [Snapshot] (スナップショット) セクションが更新され、[% Object Lock bytes] (% オブジェクトロックバイト) メトリクスを含むデータ保護メトリクスが表示されます。アカウントまたは組織のオブジェクトロックバイトの全体のパーセンテージを確認できます。
5. バケットあたりの [% Object Lock bytes] (% オブジェクトロックバイト) を確認するには、[Top N overview] (トップ N の概要) セクションまでスクロールしてください。

   オブジェクトロックのオブジェクトレベルのデータを取得するには、[Object Lock object count] (オブジェクトロックオブジェクト数) と[% Object Lock objects] (% オブジェクトロックオブジェクト) メトリクスを使用することもできます。
6. [Metric] (メトリクス) には、[Data protection] (データ保護) カテゴリから [% Object Lock bytes] (% オブジェクトロックバイト) を選択します。

   デフォルトでは、[Top N overview for date] (日付のトップ N の概要) セクションには、上位 3 つのバケットのメトリクスが表示されます。[Top N] (トップ N) のフィールドでバケット数を増やすことができます。[Top N overview for date] (日付のトップ N の概要) セクションには、前日または前週からの変化率と、傾向を視覚化するスパークラインも表示されます。この傾向は、無料メトリクスの場合は 14 日間、高度なメトリクスとレコメンデーションの場合は 30 日間の傾向です。

   > **Note**
   >
   > S3 ストレージレンズの高度なメトリクスとレコメンデーションの場合、メトリクスは 15 か月間クエリで利用できます。詳細については、「メトリクスの選択」を参照してください。
7. [% Object Lock bytes] (% オブジェクトロックバイト) については、次のデータを確認してください。

   - [Top number accounts] (上位アカウント数) – [% Object Lock bytes] (% オブジェクトロックバイト) が最も高いアカウントと最も低いアカウントを確認できます。
   - [Top number Regions] (上位リージョン数) – リージョン別の [% Object Lock bytes] (% オブジェクトロックバイト) の内訳を表示します。

- [Top number buckets] (上位バケット数) – [% Object Lock bytes] (% オブジェクトロックバイト) が最も高いバケットと最も低いバケットを確認できます。

オブジェクト所有権設定を監査するための S3 の使用

Amazon S3 オブジェクト所有権は、S3 バケットレベルの設定で、アクセスコントロールリスト (ACL) を無効にし、バケット内のオブジェクトの所有権を制御するために使用できます。オブジェクト所有権をバケット所有者強制に設定する場合、アクセスコントロールリスト (ACL) を無効にして、バケット内のすべてのオブジェクトの所有権を取得できます。この方法により、Amazon S3 に保存されているデータのアクセス管理を簡素化できます。

デフォルトでは、別の AWS アカウント がオブジェクトを S3 バケットにアップロードすると、そのアカウント (オブジェクトライター) がオブジェクトを所有し、そのオブジェクトにアクセスでき、ACL を介して他のユーザーにそのオブジェクトへのアクセスを許可できます。オブジェクトの所有権を使用して、このデフォルトの動作を変更できます。

Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。そのため、オブジェクトごとに個別に制御する必要がある異常な状況を除き、ACL を無効にすることをお勧めします。オブジェクトの所有権をバケット所有者に強制設定することで、ACL を無効にして、アクセスコントロールに関するポリシーに依存できます。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

S3 ストレージレンズのアクセス管理メトリクスを使用すると、ACL が無効になっていないバケットを特定できます。これらのバケットを特定したら、ACL のアクセス許可をポリシーに移行し、これらのバケットの ACL を無効にできます。

トピック

- ステップ 1: オブジェクト所有権設定の一般的な傾向を特定する
- ステップ 2: オブジェクト所有権設定のバケットレベルの傾向を把握する
- ステップ 3: オブジェクト所有権の設定をバケット所有者強制に更新し、ACL を無効にする

ステップ 1: オブジェクト所有権設定の一般的な傾向を特定する

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。

3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. [Snapshot for date] (日付のスナップショット) セクションの [Metrics categories] (メトリクスカテゴリ) で、[Access management] (アクセス管理) を選択します。

   [Snapshot for date] (日付のスナップショット) セクションが更新され、[% Object Ownership bucket owner enforced] (% オブジェクト所有権のバケット所有者強制) メトリクスが表示されます。アカウントまたは組織内のバケットで、オブジェクト所有権のバケット所有者強制設定を使用して ACL を無効にしているバケット全体の割合を確認することができます。

ステップ 2: オブジェクト所有権設定のバケットレベルの傾向を把握する

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードを選択します。
4. バケットレベルの詳細なメトリクスを表示するには、[Bucket] (バケット) タブを選択します。
5. [Distribution by buckets for date] (日付のバケットによる配分) セクションで、[% Object Ownership bucket owner enforced] (% オブジェクト所有権のバケット所有者強制) メトリクスを選択します。

   グラフが更新され、[% Object Ownership bucket owner enforced].(% オブジェクト所有権のバケット所有者強制) のバケットごとの内訳が表示されます。どのバケットがオブジェクト所有権のバケット所有者強制設定を使用して ACL を無効にしているかを確認できます。
6. バケット所有者強制設定をコンテキストで確認するには、[Buckets] (バケット) セクションまでスクロールします。[Metrics categories] (メトリクスカテゴリ) で、[Access management] (アクセス管理) を選択します。次に、[Summary] (概要) を選択解除します。

   [Buckets] (バケット) リストには、バケット所有者強制、バケット所有者優先、オブジェクト作成者の 3 つのオブジェクト所有権設定すべてのデータが表示されます。
7. [Buckets] (バケット) リストをフィルタリングして特定のオブジェクト所有権設定のみのメトリクスを表示するには、設定アイコン () を選択します。
8. 表示しないメトリクスを選択解除します。

9. (オプション) [Page size] (ページサイズ) で、リストに表示するバケットの数を選択します。
10. [Confirm] (確認) を選択します。

ステップ 3: オブジェクト所有権の設定をバケット所有者強制に更新し、ACL を無効にする

オブジェクト所有権にオブジェクト作成者およびバケット所有者優先設定を使用しているバケットを特定したら、ACL アクセス許可をバケットポリシーに移行できます。ACL アクセス許可の移行が完了したら、オブジェクト所有権の設定をバケット所有者強制に更新して ACL を無効にできます。詳細については、「ACL を無効にする前提条件」を参照してください。

S3 ストレージレンズのメトリクスを使用してパフォーマンスを改善する

S3 ストレージレンズの高度なメトリクスを有効にしている場合は、詳細なステータスコードのメトリクスを使用して、リクエストの成功または失敗の数を取得できます。この情報は、アクセスまたはパフォーマンス問題のトラブルシューティングに役立てることができます。詳細なステータスコードのメトリクスには、「403 Forbidden」や「503 Service Unavailable」などの HTTP ステータスコードの数が表示されます。S3 バケット、アカウント、組織にわたる詳細なステータスコードのメトリクスの全体的な傾向を調べることができます。次に、バケットレベルのメトリクスを詳しく調べて、そのバケットに現在アクセスしていてエラーの原因となっているワークロードを特定できます。

例えば、403 Forbidden エラー数メトリクスを見ると、適切なアクセス権限が適用されていない状態でバケットにアクセスしているワークロードを特定できます。これらのワークロードを特定したら、S3 ストレージレンズ以外で詳しく調べて 403 Forbidden エラーのトラブルシューティングを行うことができます。

この例では、403 Forbidden エラー数と % 403 Forbidden エラーのメトリクスを使用して、403 Forbidden エラーの傾向分析を行う方法を示しています。これらのメトリクスを使用すると、適切なアクセス権限が適用されていないバケットにアクセスしているワークロードを特定できます。他の詳細なステータスコードのメトリクスについても同様の傾向分析を行うことができます。詳細については、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

前提条件

S3 ストレージレンズのダッシュボードに詳細なステータスコードのメトリクスを表示するには、S3 ストレージレンズの [Advanced metrics and recommendations] (高度なメトリクスとレコメンデーション) を有効にしてから、[Detailed status code metrics] (詳細なステータスコードのメトリクス) を選択する必要があります。詳細については、「Amazon S3 Storage Lens ダッシュボードの作成と更新」を参照してください。

トピック

ステップ 1: 個々の HTTP ステータスコードの傾向分析を行う

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードの名前を選択します。
4. [Trends and distributions] (傾向とディストリビューション) セクションの [Primary metric] (プライマリメトリクス) で、[Detailed status codes] (詳細なステータスコード) カテゴリから [403 Forbidden error count] (403 Forbidden エラー数) を選択します。[Secondary metric] (セカンダリメトリクス) で、[% 403 Forbidden errors] (% 403 Forbidden エラー) を選択します。
5. [Top N overview for date] (日付の上位 N の概要) セクションまで下にスクロールします。[Metrics] (メトリクス) では、[Detailed status codes] (詳細なステータスコード) カテゴリから [403 Forbidden error count] (403 Forbidden エラー数) または [% 403 Forbidden errors] (% 403 Forbidden エラー)を選択します。

   [Top N overview for date] (日付の上位 N の概要) セクションが更新され、アカウント、AWS リージョン、バケット別の上位の 403 Forbidden エラー数が表示されます。

ステップ 2: バケットごとのエラー数を分析する

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [Dashboards] (ダッシュボード) リストで、表示するダッシュボードの名前を選択します。
4. ストレージレンズダッシュボードで、[Bucket] (バケット) タブを選択します。
5. [Buckets] (バケット) セクションまで下にスクロールします。[Metrics categories] (メトリクスのカテゴリ) で、[Detailed status codes] (詳細なステータスコード) のメトリクスを選択します。次に、[Summary] (概要) を選択解除します。

[Bucket] (バケット) リストが更新され、詳細なステータスコードの利用可能なメトリクスがすべて表示されます。この情報を使用して、特定の HTTP ステータスコードの割合が高いバケットと、バケット間で共通するステータスコードを確認できます。

6. [Bucket] (バケット) リストをフィルタリングして特定の詳細なステータスコードのメトリクスのみを表示するには、設定アイコン () を選択します。
7. [Bucket] (バケット) リストに表示したくない詳細なステータスコードのメトリクスがある場合は、トグルをオフにします。
8. (オプション) [Page size] (ページサイズ) で、リストに表示するバケットの数を選択します。
9. [Confirm] (確認) を選択します。

   [Bucket] (バケット) リストには、指定したバケットの数のエラー数メトリクスが表示されます。この情報を使用して、多くのエラーが発生している特定のバケットを特定し、バケットごとにエラーをトラブルシューティングできます。

ステップ 3: エラーをトラブルシューティングする

特定の HTTP ステータスコードの割合が高いバケットを特定したら、これらのエラーをトラブルシューティングできます。詳細については、次を参照してください。

- Amazon S3 でファイルをアップロードしようとすると、「403 Forbidden」(403 禁止) エラーが表示されるのはなぜですか?
- Amazon S3 のバケットポリシーを変更しようとすると、「403 Forbidden」(403 禁止) エラーが表示されるのはなぜですか?
- すべてのリソースが同じ AWS アカウント に属している Amazon S3 バケットの「403 Forbidden」(403 禁止) エラーをトラブルシューティングするにはどうすればよいですか?
- Amazon S3 からの HTTP 500 または 503 エラーをトラブルシューティングするにはどうすればよいですか?

Amazon S3 Storage Lens のメトリクスに関する用語集

Amazon S3 ストレージレンズメトリクスの用語集には、S3 Storage Lensの無料および高度なメトリクスの完全なリストが提供されています。

S3 ストレージレンズでは、すべてのダッシュボードと構成に対して無料のメトリクスを提供していますが、さらに、高度なメトリクスにアップグレードできるオプションも用意されています。

- 無料のメトリクスには、アカウント内のバケットやオブジェクトの数など、ストレージ使用状況に関連するメトリクスが含まれています。無料のメトリクスには、コスト最適化メトリクスやデータ保護メトリクスなどのユースケースに基づくメトリクスも含まれています。すべての無料メトリクスは毎日収集され、データは最長 14 日間のクエリに使用できます。
- 高度なメトリクスとレコメンデーションには、無料のメトリクスに含まれるすべてのメトリクスに加えて、高度なデータ保護やコスト最適化メトリクスなどの追加のメトリクスが含まれます。高度なメトリクスには、アクティビティメトリクスや詳細なステータスコードメトリクスなど、その他のメトリクスカテゴリーも含まれます。高度なメトリクスのデータは 15 か月間クエリで使用できます。

  S3 Storage Lens で高度な推奨のメトリクスを使用すると、追加料金が発生します。詳細については、「Amazon S3 の料金」を参照してください。高度なメトリクスとレコメンデーションの詳細については、「メトリクスの選択」を参照してください。

> **Note**
>
> Storage Lens グループでは、無料利用枠のストレージメトリクスしか使用できません。高度なメトリクスは、Storage Lens グループレベルで利用できません。

メトリクス名

次の表のメトリクス名列には、S3 コンソールの各 S3 ストレージレンズの名前が表示されます。CloudWatch とエクスポート列には、Amazon CloudWatch の各メトリクスの名前と、S3 ストレージレンズダッシュボードで設定できるメトリクスエクスポートファイルが表示されます。

派生メトリクスの式

派生メトリクスは、メトリクスのエクスポートおよび CloudWatch 公開オプションでは使用できません。ただし、派生メトリクスの式列に表示されているメトリクス式を使用してコンピューティングすることは可能です。

Amazon S3 ストレージレンズメトリクスの単位倍数プレフィックス記号 (K、M、G など) の解釈

Amazon S3 ストレージレンズメトリクスの単位倍数はプレフィックス記号で書き込まれます。これらのプレフィックス記号は、国際度量衡局 (BIPM) によって標準化されている国際単位系 (SI) 記号と

一致しています。また、これらの記号は、測定単位の統一コード (UCUM) でも使用されています。詳細については、「SI プレフィックス記号の一覧」を参照してください。

> **Note**
>
> - S3 ストレージバイトの測定単位はバイナリギガバイト (GB) で、1 GB は 2^{30} バイト、1 TB は 2^{40} バイト、1 PB は 2^{50} バイトです。この測定単位は、国際電気標準会 (IEC) で定義されているように、ギビバイト (GiB) とも呼ばれます。
> - ライフサイクル設定に基づいて、オブジェクトが有効期限に達すると、存続期間が終了したオブジェクトは自動的に Amazon S3 削除キューに追加され、非同期的に削除されます。そのため、有効期限が切れる日と Amazon S3 がオブジェクトを削除する日との間に遅延が生じることがあります。S3 Storage Lens には、有効期限が切れても削除されていないオブジェクトのメトリクススは含まれません。S3 ライフサイクルの有効期限アクションの詳細については、「オブジェクトの有効期限」を参照してください。

S3 ストレージレンズのメトリクスに関する用語集

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
合計ストレージ	StorageBytes	不完全なマルチパートアップロード、オブジェクトメタデータ、削除マーカーを含む、合計ストレージ	空き	まとめ	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Object count (オブジェクト数)	ObjectCount	オブジェクトの総数	空き	まとめ	N	-
平均のオブジェクトサイズ	-	平均のオブジェクトサイズ	空き	まとめ	Y	sum(StorageBytes)/sum(ObjectCount)
Active buckets (アクティブなバケット)	-	ストレージが 0 バイトを超える、アクティブに使用されているバケットの総数	空き	まとめ	Y	-
バケット	-	バケットの合計数	空き	まとめ	Y	-
アカウント	-	ストレージがスコープ内にあるアカウントの数	空き	まとめ	Y	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Current version bytes (現在のバージョンのバイト数)	CurrentVersionStorageBytes	オブジェクトの現在のバージョンのバイト数	空き	コスト最適化	N	-
% Current Version Bytes (現在のバージョンのバイト数)	-	オブジェクトの現在のバージョンであるスコープ内のバイトの割合	空き	コスト最適化	Y	sum(CurrentVersionStorageBytes)/sum(StorageBytes)
Current version object count (現在のバージョンのオブジェクト数)	現在のバージョンのオブジェクト数	現在のバージョンのオブジェクトの数	空き	コスト最適化	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% current version objects (現在のバージョンのオブジェクト)	-	スコープ内のオブジェクトのうち、現在のバージョンであるオブジェクトの割合	空き	コスト最適化	Y	sum(CurrentVersionObjectCount)/sum(ObjectCount)
Noncurrent version bytes (現在のバージョンではないバイト数)	現在のバージョンではないストレージバイト数	現在のバージョンではないバイト数	空き	コスト最適化	N	-
% noncurrent version bytes (現在のバージョンではないバイト数)	-	スコープ内で、現在のバージョンではないバイトの割合	空き	コスト最適化	Y	sum(NonCurrentVersionStorageBytes)/sum(StorageBytes)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Noncurrent version object count (現在のバージョンではないオブジェクト数)	現在のバージョンではないオブジェクト数	現在のバージョンではないオブジェクトの数	空き	コスト最適化	N	-
% noncurrent version objects (現在のバージョンではないオブジェクト数)	-	スコープ内のオブジェクトのうち、現在のバージョンではないオブジェクトの割合	空き	コスト最適化	Y	sum(NonCurrentVersionObjectCount)/sum(ObjectCount)
Delete marker bytes (削除マーカーのバイト数)	DeleteMarkerStorageBytes	削除マーカーとなるスコープ内のバイト数	空き	コスト最適化	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% delete marker bytes (削除マーカーバイト数)	-	削除マーカーであるバイト数の、スコープ内での割合	空き	コスト最適化	Y	sum(DeleteMarkerStorageBytes)/ sum(StorageBytes)
Delete marker object count (削除マーカーオブジェクト数)	DeleteMarkerObjectCount 削除マーカーオブジェクト数	削除マーカーがあるオブジェクトの総数	空き	コスト最適化	N	-
% delete marker objects (削除マーカーオブジェクト)	-	削除マーカーのあるオブジェクトの、スコープ内での割合	空き	コスト最適化	Y	sum(DeleteMarkerObjectCount)/ sum(ObjectCount)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Incomplete multipart upload bytes (未完了のマルチパートアップロードのバイト数)	不完全なマルチパートアップロードストレージバイト数	スコープ内で未完了のマルチパートアップロードの合計バイト数	空き	コスト最適化	N	-
% incomplete multipart upload bytes (未完了のマルチパートアップロードのバイト数)	-	未完了のマルチパートアップロードの結果であるバイト数の、スコープ内での割合	空き	コスト最適化	Y	sum(IncompleteMultipartUploadStorageBytes)/ sum(StorageBytes)
Incomplete multipart upload object count (未完了のマルチパートアップロードのオブジェクト数)	MultipartUploadObjectCount 不完全なマルチパートアップロードストレージオブジェクト数	スコープ内でマルチパートアップロードが不完全であるオブジェクトの数	空き	コスト最適化	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% incomplete multipart upload objects (未完了のマルチパートアップロードオブジェクト)	-	マルチパートアップロードが不完全であるオブジェクト数の、スコープ内での割合	空き	コスト最適化	Y	sum(IncompleteMultipartUploadObjectCount)/sum(ObjectCount)
Incomplete multipart upload storage bytes greater than 7 days old (7 日以上経過した未完了のマルチパートアップロードストレージバイト数)	IncompleteMPUStorageBytesOlderThan7Days	7 日以上経過した未完了のマルチパートアップロードのスコープ内の合計バイト数	空き	コスト最適化	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% incomplete multipart upload storage bytes greater than 7 days old (7 日以上経過した未完了のマルチパートアップロードストレージバイト)	-	7 日以上経過した未完了のマルチパートアップロードのバイトの割合	空き	コスト最適化	Y	sum(IncompleteMPUStorageBytesOlderThan7Days)/sum(StorageBytes)
Incomplete multipart upload object count greater than 7 days old (7 日以上経過した未完了のマルチパートアップロードオブジェクト数の割合)	IncompleteMPUObjectCountOlderThan7Days	7 日以上経過した未完了のマルチパートアップロードであるオブジェクトの数	空き	コスト最適化	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% incomplete multipart upload object count greater than 7 days old (7 日以上経過した未完了のマルチパートアップロードオブジェクト数)	-	7 日以上経過した未完了のマルチパートアップロードであるオブジェクトの割合	空き	コスト最適化	Y	sum(IncompleteMPUObjectCountOlderThan7Days)/sum(ObjectCount)
Transition lifecycle rule count (移行ライフサイクルルールの数)	TransitionLifecycleRuleCount	オブジェクトを別のストレージクラスに移行するためのライフサイクルルールの数	アドバンスト	コスト最適化	N	-
Average transition lifecycle rules per bucket (バケットあたりの平均移行ライフサイクルルール)	-	オブジェクトを別のストレージクラスに移行するライフサイクルルールの平均数	アドバンスト	コスト最適化	Y	sum(TransitionLifecycleRuleCount)/sum(DistinctNumberOfBuckets)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Expiration lifecycle rule count (有効期限切れのライフサイクルルールの数)	ExpirationLifecycleRuleCount	オブジェクトを期限切れにするライフサイクルルールの数	アドバンスト	コスト最適化	N	-
Average expiration lifecycle rules per bucket (バケットあたりの平均有効期限ライフサイクルルール)	-	オブジェクトを期限切れにするライフサイクルルールの平均数	アドバンスト	コスト最適化	Y	sum(ExpirationLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
Noncurrent version transition lifecycle rule count (最新バージョン移行ライフサイクルルール数)	NoncurrentVersionTransitionLifecycleRuleCount	最新でないオブジェクトバージョンを別のストレージクラスに移行するためのライフサイクルルールの数	アドバンスト	コスト最適化	N	

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Average noncurrent version transition lifecycle rules per bucket (バケットごとの非最新バージョン移行ライフサイクルルールの平均数)	-	最新でないオブジェクトバージョンを別のストレージクラスに移行するためのライフサイクルルールの平均数	アドバンスト	コスト最適化	Y	sum(NoncurrentVersionTransitionLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
Noncurrent version expiration lifecycle rule count (最新バージョンの有効期限切れライフサイクルルール数)	NoncurrentVersionExpirationLifecycleRuleCount	最新でないオブジェクトバージョンを期限切れにするライフサイクルルールの数	アドバンスト	コスト最適化	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Average noncurrent version expiration lifecycle rules per bucket (バケットごとの最新でないバージョンの有効期限ライフサイクルルールの平均数)	-	最新でないオブジェクトバージョンを期限切れにするライフサイクルルールの平均数	アドバンスト	コスト最適化	Y	sum(NoncurrentVersionExpirationLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
Abort incomplete multipart upload lifecycle rule count (未完了のマルチパートアップロードのライフサイクルルール数の中止	AbortIncompleteMPULifecycleRuleCount	未完了のマルチパートアップロードを削除するライフサイクルルールの数	アドバンスト	コスト最適化	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Average abort incomplete multipart upload lifecycle rules per bucket (バケットあたりの未完了のマルチパートアップロードライフサイクルルールの平均中止数)	-	未完了のマルチパートアップロードを削除するライフサイクルルールの平均数	アドバンスト	コスト最適化	Y	sum(AbortIncompleteMPULifecycleRuleCount)/sum(DistinctNumberOfBuckets)
Expired object delete marker lifecycle rule count (期限切れオブジェクト削除マーカーのライフサイクルルール数)	ExpiredObjectDeleteMarkerLifecycleRuleCount	期限切れオブジェクト削除マーカーを削除するライフサイクルルールの数	アドバンスト	コスト最適化	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Average expired object delete marker lifecycle rules per bucket (バケットあたりの期限切れオブジェクト削除マーカーのライフサイクルルールの平均値)	-	期限切れオブジェクト削除マーカーを削除するライフサイクルルールの平均数	アドバンスト	コスト最適化	Y	sum(ExpiredObjectDeleteMarkerLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
Total lifecycle rule count (ライフサイクルルールの総数)	TotalLifecycleRuleCount	ライフサイクルルールの総数	アドバンスト	コスト最適化	N	-
Average lifecycle rule count per bucket (バケットあたりの平均ライフサイクルルール数)	-	ライフサイクルルールの平均数	アドバンスト	コスト最適化	Y	sum(TotalLifecycleRuleCount)/sum(DistinctNumberOfBuckets)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Encrypted bytes (暗号化されたバイト数)	暗号化ストレージバイト数	暗号化されたバイトの合計数	空き	データ保護	N	-
% encrypted bytes (暗号化されたバイト数)	-	暗号化された合計バイト数の割合	空き	データ保護	Y	sum(EncryptedObjectCount)/sum(StorageBytes)
Encrypted object count (暗号化されたオブジェクトの数)	EncryptedObjectCount	暗号化されたオブジェクトの合計数	空き	データ保護	N	-
% encrypted objects (暗号化されたオブジェクト数)	-	暗号化されたオブジェクトの割合	空き	データ保護	Y	sum(EncryptedStorageBytes)/sum(ObjectCount)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Unencrypted bytes (暗号化されていないバイト数)	UnencryptedStorageBytes	暗号化されていないバイト数	空き	データ保護	Y	sum(StorageBytes) - sum(EncryptedStorageBytes)
% unencrypted bytes (暗号化されていないバイト数)	-	暗号化されていないバイト数の割合	空き	データ保護	Y	sum(UnencryptedStorageBytes)/ sum(StorageBytes)
Unencrypted object count (暗号化されていないオブジェクト数)	UnencryptedObjectCount	暗号化されていないオブジェクトの総数	空き	データ保護	Y	sum(ObjectCount) - sum(EncryptedObjectCount)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% unencrypted objects (暗号化されていないオブジェクト数)	-	暗号化されていないオブジェクトの割合	空き	データ保護	Y	sum(UnencryptedStorageBytes)/ sum(ObjectCount)
Replicated storage bytes source (レプリケートされたストレージバイト数ソース)	ReplicatedStorageBytesSource	ソースバケットからレプリケートされた合計バイト数	空き	データ保護	N	-
% replicated bytes source (レプリケートされたバイトソース)	-	ソースバケットからレプリケートされた合計バイト数の割合	空き	データ保護	Y	sum(ReplicatedStorageBytesSource)/ sum(StorageBytes)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Replicated object count source (レプリケートされたオブジェクト数のソース)	ReplicatedObjectCountSource	ソースバケットからレプリケートされたオブジェクトの数	空き	データ保護	N	-
% replicated objects source (レプリケートされたオブジェクトソース)	-	ソースバケットからレプリケートされた全オブジェクト数の割合	空き	データ保護	Y	sum(ReplicatedStorageObjectCount)/sum(ObjectCount)
Replication storage bytes destination (レプリケーションストレージのバイト数送信先)	レプリケートされたストレージバイト数	送信先バケットにレプリケートされた合計バイト数	空き	データ保護	Y	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% replicated bytes destination (レプリケートされるバイト数送信先)	-	送信先バケットにレプリケートされた合計バイト数の割合	空き	データ保護	Y	sum(ReplicatedStorageBytesSource)/ sum(StorageBytes)
Replicated object count destination (レプリケートされたオブジェクト数の送信先)	レプリケートされたオブジェクト数	送信先バケットにレプリケートされたオブジェクトの数	空き	データ保護	Y	-
% replicated objects destination (レプリケートされたオブジェクトの送信先)	-	送信先バケットにレプリケートされた全オブジェクト数の割合	空き	データ保護	Y	sum(ReplicatedObjectCount)/ sum(ObjectCount)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Object Lock bytes (オブジェクトロックバイト数)	オブジェクトロックが有効なストレージバイト数	オブジェクトロックが有効なストレージバイトの総数	空き	データ保護	Y	sum(UnencryptedStorageBytes)/ sum(ObjectLockEnabledStorageCount) - sum(ObjectLockEnabledStorageBytes)
% Object Lock bytes (オブジェクトロックバイト数)	-	オブジェクトロックが有効なストレージバイトの割合	空き	データ保護	Y	sum(ObjectLockEnabledStorageBytes)/ sum(StorageBytes)
Object Lock object count (オブジェクトロックオブジェクト数)	不完全なマルチパートアップロードストレージオブジェクト数	オブジェクトロックオブジェクトの総数	空き	データ保護	Y	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% Object Lock objects (オブジェクトロックオブジェクト数)	-	オブジェクトロックが有効になっている全オブジェクト数の割合	空き	データ保護	Y	sum(ObjectLockEnabledObjectCount)/sum(ObjectCount)
Versioning-enabled bucket count (バージョニングが有効なバケット数)	VersioningEnabledBucketCount	S3 バージョニングが有効になっているバケットの数	空き	データ保護	N	-
% versioning-enabled buckets (バージョニングが有効なバケット)	-	S3 バージョニングが有効になっているバケットの割合	空き	データ保護	Y	sum(VersioningEnabledBucketCount)/sum(DistinctNumberOfBuckets)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
MFA delete-enabled bucket count (MFA Delete が有効なバケット数)	MFADeleteEnabledBucketCount	MFA (多要素認証) Delete が有効になっているバケット数	空き	データ保護	N	-
% MFA delete-enabled buckets (MFA Delete が有効化されたバケット)	-	MFA (多要素認証) Delete が有効化されたバケットの割合	空き	データ保護	Y	sum(MFADeleteEnabledBucketCount)/sum(DistinctNumberOfBuckets)
SSE-KMS enabled bucket count (SSE-KMS が有効化されたバケット数)	SSEKMSEnabledBucketCount	デフォルトのバケット暗号化に AWS Key Management Service キーのサーバー側暗号化 (SSE-KMS) を使用しているバケット数	空き	データ保護	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% SSE-KMS enabled buckets (SSE-KMS が有効化されたバケット)	-	デフォルトのバケット暗号化に SSE-KMS を使用しているバケットの割合	空き	データ保護	Y	sum(SSEKMSEnabledBucketCount)/ sum(DistinctNumberOfBuckets)
All unsupported signature requests (サポートされていないすべての署名リクエスト)	AllUnsupportedSignatureRequests	サポートされていない AWS 署名バージョンを使用したリクエストの総数	アドバンスト	データ保護	N	-
% all unsupported signature requests (サポートされていないすべての署名リクエスト)	-	サポートされていない AWS 署名バージョンを使用するリクエストの割合	アドバンスト	データ保護	Y	sum(AllUnsupportedSignatureRequests) / sum(AllRequests)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
All unsupported TLS requests (サポートされていないすべての TLS リクエスト)	AllUnsupportedTLSRequests	サポートされていない Transport Layer Security (TLS) を使用するリクエストの数	アドバンスト	データ保護	N	-
% all unsupported TLS requests (サポートされていないすべての TLS リクエスト)	-	サポートされていない TLS バージョンを使用するリクエストの割合	アドバンスト	データ保護	Y	sum(AllUnsupportedTLSRequests)/sum(AllRequests)
All SSE-KMS requests (すべての SSE-KMS リクエスト)	AllSSEKMSRequests	SSE-KMS を指定したリクエストの総数	アドバンスト	データ保護	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% all SSE-KMS requests (すべての SSE-KMS リクエスト)	-	SSE-KMS を指定したリクエストの割合	アドバンスト	データ保護	Y	sum(AllSSEKMSRequests)/sum(AllRequests)
Same-Region Replication rule count (同一リージョンレプリケーションルールの数)	SameRegionReplicationRuleCount	同一リージョンレプリケーション (SRR) のレプリケーションルールの数	アドバンスト	データ保護	N	-
Average Same-Region Replication rules per bucket (バケットあたりの同一リージョンレプリケーションルールの平均数)	-	SRR の平均レプリケーションルールの数	アドバンスト	データ保護	Y	sum(SameRegionReplicationRuleCount)/sum(DistinctNumberOfBuckets)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Cross-Region Replication rule count (クロスリージョンレプリケーションルールの数)	CrossRegionReplicationRuleCount	クロスリージョンレプリケーション (CRR) のレプリケーションルールの数	アドバンスト	データ保護	N	-
Average Cross-Region Replication rules per bucket (バケットあたりのクロスリージョンレプリケーションの平均数)	-	CRR のレプリケーションルールの平均数	アドバンスト	データ保護	Y	sum(CrossRegionReplicationRuleCount) / sum(DistinctNumberOfBuckets)
Same-account replication rule count (同一アカウントレプリケーションルールの数)	SameAccountReplicationRuleCount	同じアカウント内のレプリケーション用のレプリケーションルールの数	アドバンスト	データ保護	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Average same-account replication rules per bucket (バケットあたりの同一アカウントレプリケーションルール平均数)	-	同じアカウント内のレプリケーション用のレプリケーションルールの平均数	アドバンスト	データ保護	Y	sum(SameAccountReplicationRuleCount) / sum(DistinctNumberOfBuckets)
Cross-account replication rule count (クロスアカウントレプリケーションルールの数)	CrossAccountReplicationRuleCount	クロスアカウントレプリケーションのレプリケーションルールの数	アドバンスト	データ保護	N	-
Average cross-account replication rules per bucket (バケットあたりのクロスアカウントレプリケーションルールの平均数)	-	クロスアカウントレプリケーションのレプリケーションルールの平均数	アドバンスト	データ保護	Y	sum(CrossAccountReplicationRuleCount)/ sum(DistinctNumberOfBuckets)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Invalid destination replication rule count (無効な送信先レプリケーションルールの数)	InvalidDestinationReplicationRuleCount	レプリケーション先が有効でないレプリケーションルールの数	アドバンスト	データ保護	N	-
Average invalid destination replication rules per bucket (バケットあたりの無効な送信先レプリケーションルールの平均数)	-	レプリケーション先が有効でないレプリケーションルールの平均数	アドバンスト	データ保護	Y	sum(InvalidReplicationRuleCount)/ sum (Distinct NumberOfBuckets)
Total replication rule count (レプリケーションルールの総数)	-	レプリケーションルールの総数	アドバンスト	データ保護	Y	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Average replication rule count per bucket (バケットあたりの平均レプリケーションルール数)	-	レプリケーションルールの合計数の平均	アドバンスト	データ保護	Y	sum(all replication rule count metrics)/ sum(DistinctNumberOfBuckets)
Object Ownership bucket owner enforced bucket count (オブジェクトの所有権のバケット所有者強制バケット数)	ObjectOwnershipBucketOwnerEnforcedBucketCount	オブジェクトの所有権のバケット所有者強制設定を使用して、アクセスコントロールリスト (ACL) を無効にしたバケットの総数	空き	アクセス管理	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% Object Ownership bucket owner enforced buckets (オブジェクトの所有権のバケット所有者強制バケット)	-	オブジェクトの所有権のバケット所有者強制設定を使用して ACL を無効にしたバケットの割合	空き	アクセス管理	Y	sum(ObjectOwnershipBucketOwnerEnforcedBucketCount)/ sum (DistinctNumberOfBuckets)
Object Ownership bucket owner preferred bucket count (オブジェクトの所有権のバケット所有者の優先バケット数)	ObjectOwnershipBucketOwnerPreferredBucketCount	オブジェクト所有権のバケット所有者優先設定を使用するバケットの総数	空き	アクセス管理	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% Object Ownership bucket owner preferred buckets (オブジェクトの所有権のバケット所有者優先バケット)	-	オブジェクトの所有権のバケット所有者優先設定を使用するバケットの割合	空き	アクセス管理	Y	sum(ObjectOwnershipBucketOwnerPreferredBucketCount)/ sum(DistinctNumberOfBuckets)
Object Ownership object writer bucket count (オブジェクトの所有権オブジェクトライターバケット数)	ObjectOwnershipObjectWriterBucketCount	オブジェクトの所有権のオブジェクトライター設定を使用するバケットの総数	空き	アクセス管理	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% Object Ownership object writer buckets (オブジェクトの所有権のオブジェクトライターバケット)	-	オブジェクト所有権のオブジェクトライター設定を使用するバケットの割合	空き	アクセス管理	Y	sum(ObjectOwnershipObjectWriterBucketCount)/sum(DistinctNumberOfBuckets)
Transfer Acceleration enabled bucket count (Transfer Acceleration が有効化されたバケット数)	TransferAccelerationEnabledBucketCount	Transfer Acceleration が有効化されたバケットの総数	空き	パフォーマンス	N	-
% Transfer Acceleration enabled buckets (Transfer Acceleration が有効化されたバケット)	-	Transfer Acceleration が有効化されたバケットの割合	空き	パフォーマンス	Y	sum(TransferAccelerationEnabledBucketCount)/sum(DistinctNumberOfBuckets)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Event Notification enabled bucket count (イベント通知が有効化されたバケット数)	EventNotificationEnabledBucketCount	イベント通知が有効化されたバケットの総数	空き	のイベント	N	
% Event Notification enabled buckets (イベント通知が有効化されたバケット)	-	イベント通知が有効化されたバケットの割合	空き	のイベント	Y	sum(EventNotificationEnabledBucketCount)/sum(DistinctNumberOfBuckets)
すべてのリクエスト	AllRequests	作成されたリクエストの総数	アドバンスト	アクティビティ	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
リクエストの取得	GetRequests	作成された GET リクエストの総数	アドバンスト	アクティビティ	N	-
Put requests (Put リクエスト)	PutRequests	作成された PUT リクエストの総数	アドバンスト	アクティビティ	N	-
Head requests (Head リクエスト)	HeadRequests	作成された HEAD リクエストの総数	アドバンスト	アクティビティ	N	-
Delete requests (Delete リクエスト)	DeleteRequests	作成された DELETE リクエストの総数	アドバンスト	アクティビティ	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
リクエストを一覧表示	ListRequests	作成された LIST リクエストの総数	アドバンスト	アクティビティ	N	-
Post requests (Post リクエスト)	PostRequests	作成された POST リクエストの総数。	アドバンスト	アクティビティ	N	-
Select requests (Select リクエスト)	SelectRequests	S3 Select リクエストの総数	アドバンスト	アクティビティ	N	-
Select scanned bytes (スキャンされた Select バイト)	SelectScannedBytes	スキャンされた S3 Select バイトの数	アドバンスト	アクティビティ	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
Select returned bytes (返された Select バイト)	SelectReturnedBytes	返された S3 Select バイトの数	アドバンスト	アクティビティ	N	-
ダウンロードされたバイト数	BytesDownloaded	ダウンロードされたバイトの数	アドバンスト	アクティビティ	N	-
% retrieval rate (取得率)	-	ダウンロードされたバイト数の割合	アドバンスト	アクティビティ	Y	sum(BytesDownloaded)/sum(StorageBytes)
アップロードされたバイト数	BytesUploaded	アップロード済みとなっているバイト数。	アドバンスト	アクティビティ	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% ingest ratio (取り込み率)	-	アップロード済みとなっているバイト数の割合	アドバンスト	アクティビティ	Y	sum(BytesUploaded) / sum(StorageBytes)
4xx errors (4xx エラー)	4xxErrors	HTTP 4xx ステータスコードの総数	アドバンスト	アクティビティ	N	-
5xx errors (5xx エラー)	5xxErrors	HTTP 5xx ステータスコードの総数	アドバンスト	アクティビティ	N	-
Total errors (エラー総数)	-	4xx と 5xx のすべてのエラーを合計した数	アドバンスト	アクティビティ	Y	sum(4xxErrors) + sum(5xxErrors)

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% error rate (エラー率)	-	4xx と 5xx のエラーの総数がリクエストの総数に占める割合	アドバンスト	アクティビティ	Y	sum(TotalErrors)/sum(TotalRequests)
200 OK status count (200 OK ステータスの数)	200OKStatusCount	200 OK ステータスコードの総数	アドバンスト	詳細なステータスコード	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% 200 OK status (200 OK ステータス)	-	200 OK ステータスコードの総数がリクエストの総数に占める割合	アドバンスト	詳細なステータスコード	Y	sum(200OKStatusCount)/sum(AllRequests)
206 Partial Content status count (206 部分コンテンツステータスの数)	206PartialContentStatusCount	206 部分コンテンツステータスコードの総数	アドバンスト	詳細なステータスコード	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% 206 Partial Content status (206 部分コンテンツステータス)	-	206 部分コンテンツステータスコードの合計数がリクエストの総数に占める割合	アドバンスト	詳細なステータスコード	Y	sum(206PartialContentStatusCount)/sum(AllRequests)
400 Bad Request error count (400 Bad Request エラー数)	400BadRequestErrorCount	400 Bad Request ステータスコードの総数	アドバンスト	詳細なステータスコード	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% 400 Bad Request errors (400 Bad Request エラー)	-	400 Bad Request ステータスコードの合計数がリクエストの総数に占める割合	アドバンスト	詳細なステータスコード	Y	sum(400BadRequestErrorCount)/sum(All Requests)
403 Forbidden error count (403 Forbidden エラー数)	403ForbiddenErrorCount	403 Forbidden ステータスコードの総数	アドバンスト	詳細なステータスコード	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% 403 Forbidden errors (403 Forbidden エラー)	-	403 Forbidden ステータスコードの総数がリクエストの総数に占める割合	アドバンスト	詳細なステータスコード	Y	sum(403ForbiddenErrorCount) / sum(AllRequests)
404 Not Found error count (404 Not Found エラー数)	404NotFoundErrorCount	404 Not Found ステータスコードの総数	アドバンスト	詳細なステータスコード	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% 404 Not Found errors (404 Not Found エラー)	-	404 Not Found ステータスコードの合計数がリクエストの総数に占める割合	アドバンスト	詳細なステータスコード	Y	sum(404NotFoundErrorCount)/sum(AllRequests)
500 Internal Server Error count (500 Internal Server Error の数)	500InternalServerErrorCount	500 Internal Server Error ステータスコードの総数	アドバンスト	詳細なステータスコード	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% 500 Internal Server Errors	-	500 Internal Server Errors の合計数がリクエストの総数に占める割合	アドバンスト	詳細なステータスコード	Y	sum(500InternalServerErrorCount)/sum(AllRequests)
503 Service Unavailable error count (503 Service Unavailable エラー数)	503ServiceUnavailableErrorCount	503 Service Unavailableステータスコードの総数	アドバンスト	詳細なステータスコード	N	-

メトリクス名	CloudWatch とエクスポート	説明	ティア[1]	カテゴリー[2]	派生	派生メトリクスの式
% 503 Service Unavailable errors (503 Service Unavailable エラー)	-	503 Service Unavailable ステータスコードの合計数がリクエスト総数に占める割合	アドバンスト	詳細なステータスコード	Y	sum(503ServiceUnavailableErrorCount) / sum(AllRequests)

[1] すべての無料ティアストレージメトリクスは、Storage Lens グループレベルで利用できます。高度なメトリクスは、Storage Lens グループレベルで利用できません。

[2] ルール数メトリクスとバケット設定メトリクスは、プレフィックスレベルで使用できません。

コンソールと API を使用した Amazon S3 ストレージレンズの使用

Amazon S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージの分析機能です。S3 ストレージレンズメトリクスを使用することで、組織全体でどれだけのストレージがあるか、または最も急速に成長しているバケットとプレフィックスは何かなどの、要約されたインサイトを生成できます。S3 ストレージレンズメトリクスを使用して、コスト最適化の機会を特定し、データ保護とセキュリティのベストプラクティスを実装し、アプリケーションワークロードのパフォーマンスを向上させることもできます。例えば、S3 ライフサイクルルールがないバケットを特定して、7 日以上経過した未完了のマルチパートアップロードを有効期限切れにできます。また、S3 レプリケーションや S3 バージョニング使用など、データ保護のベストプラクティスに従っていないバケットを特定することもできます。

また、S3 ストレージレンズは、メトリクスを分析して、ストレージコストを最適化し、データ保護に関するベストプラクティスを適用するために使用できるコンテキストに応じた推奨事項を提供します。

S3 ストレージレンズはメトリクスを集約し、Amazon S3 コンソールの [Buckets] (バケット) ページの [Account snapshot] (アカウントスナップショット) セクションにこの情報を表示します。S3 Storage Lens は、インサイトと傾向を可視化したり、外れ値にフラグ付けしたり、ストレージコストの最適化やデータ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取るために使用できるインタラクティブダッシュボードも提供します。ダッシュボードには、組織、アカウント、AWS リージョン、バケット、オブジェクト、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成して可視化できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。

以下のセクションでは、S3 Storage Lens の設定の作成、更新、表示例を示します。また、機能に関連した操作例もご覧いただけます。ここでの例には、AWS Organizations で S3 Storage Lens を使用している場合のユースケースも含まれます。この例の中の変数の値は、ご自身の用途に合った値に置き換えてご使用になれます。

トピック

コンソールでの Amazon S3 Storage Lens の使用

Amazon S3 ストレージレンズは、オブジェクトストレージの使用状況とアクティビティを組織全体で可視化するために使用できるクラウドストレージ分析機能です。S3 ストレージレンズメトリクスを使用することで、組織全体でどれだけのストレージがあるか、または最も急速に成長しているバケットとプレフィックスは何かなどの、要約されたインサイトを生成できます。S3 ストレージレンズメトリクスを使用して、コスト最適化の機会を特定し、データ保護とセキュリティのベストプラクティスを実装し、アプリケーションワークロードのパフォーマンスを向上させることもできます。例えば、S3 ライフサイクルルールがないバケットを特定して、7 日以上経過した未完了のマルチパートアップロードを有効期限切れにできます。また、S3 レプリケーションや S3 バージョニング使用など、データ保護のベストプラクティスに従っていないバケットを特定することもできます。また、S3 ストレージレンズは、メトリクスを分析して、ストレージコストを最適化し、データ保護

に関するベストプラクティスを適用するために使用できるコンテキストに応じた推奨事項を提供します。

S3 ストレージレンズはメトリクスを集約し、Amazon S3 コンソールの [Buckets] (バケット) ページの [Account snapshot] (アカウントスナップショット) セクションにこの情報を表示します。S3 Storage Lens は、インサイトと傾向を可視化したり、外れ値にフラグ付けしたり、ストレージコストの最適化やデータ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取ったりするために使用できるインタラクティブダッシュボードも提供します。ダッシュボードには、組織、アカウント、AWS リージョン、バケット、オブジェクト、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成して可視化できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。

> **Note**
>
> ダッシュボード構成への更新が正確に表示または視覚化されるまでに最大 48 時間かかる場合があります。

トピック

- Amazon S3 Storage Lens ダッシュボードの作成と更新
- Amazon S3 Storage Lens ダッシュボードの無効化と削除
- 組織レベルのダッシュボードを作成するための AWS Organizations の使用

Amazon S3 Storage Lens ダッシュボードの作成と更新

S3 ストレージレンズはメトリクスを集約し、Amazon S3 コンソールの [Buckets] (バケット) ページの [Account snapshot] (アカウントスナップショット) セクションにこの情報を表示します。S3 Storage Lens は、インサイトと傾向を可視化したり、外れ値にフラグ付けしたり、ストレージコストの最適化やデータ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取ったりするために使用できるインタラクティブダッシュボードも提供します。ダッシュボードには、組織、アカウント、AWS リージョン、バケット、オブジェクト、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成して可視化できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。

Amazon S3 Storage Lens のデフォルトのダッシュボードは、default-account-dashboard です。このダッシュボードは Amazon S3 によって事前定義されており、無料のメトリクスおよび高度なメトリ

クスをアカウント全体で集計し、その概要と傾向をコンソール上で可視化するのに役立ちます。デフォルトダッシュボードのスコープ設定を変更することはできませんが、メトリクスの選択を無料メトリクスから有料の高度なメトリクスとレコメンデーションにアップグレードしたり、オプションのメトリクスエクスポートを構成したり、デフォルトダッシュボードを無効にすることは可能です。デフォルトのダッシュボードは削除できません。

また、AWS Organizations の組織、またはアカウント内の特定のリージョンやバケットにスコープを設定できる、追加の S3 ストレージレンズカスタムダッシュボードを作成することができます。

Amazon S3 Storage Lens ダッシュボードの作成

Amazon S3 コンソールで Amazon S3 Storage Lens ダッシュボードを作成するには、次の手順に従います。

ステップ 1: ダッシュボードのスコープを定義する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ページ上部にあるナビゲーションバーで、現在表示されている AWS リージョンの名前をクリックします。次に、切り替え先のリージョンを選択します。
3. ナビゲーションペインの、[S3 ストレージレンズ] で、[ダッシュボード] を選択します。
4. [ダッシュボードの作成] を選択します。
5. ダッシュボードページの [全般] セクションで、次の操作を行います。

   a. ダッシュボードの [ホームリージョン] を表示します。ホームリージョンとは、この Storage Lens ダッシュボード設定とメトリクスが保存される AWS リージョン です。

   b. ダッシュボード名を入力します。

      ダッシュボード名は 65 文字未満で記述する必要があり、特殊文字またはスペースを含めることはできません。

      > **Note**
      >
      > ダッシュボードが作成された後は、このダッシュボード名を変更することはできません。

   c. 必要に応じて、ダッシュボードにタグを追加することもできます。タグは、ダッシュボードのアクセス許可を管理し、S3 Storage Lens のコストを追跡するために使用できます。

詳細については、IAM ユーザーガイドのリソースタグを使用したアクセスの制御、および AWS Billing ユーザーガイドの AWS 生成コスト配分タグを参照してください。

> **Note**
>
> ダッシュボードの設定には、最大 50 個のタグを追加できます。

6. [ダッシュボードスコープ] セクションで、次の操作を行います。

   a. S3 Storage Lens でダッシュボードに含める、または除外するリージョンとバケットを選択します。

   b. S3 Storage Lens で含める、または除外するバケットを、選択したリージョンの中で選択します。バケットを含めるか除外するかの設定は可能ですが、両方を同時に設定することはできません。このオプションは、組織レベルのダッシュボードを作成するときには使用できません。

> **Note**
>
> - リージョンとバケットを、含めるか除外するかの選択が可能です。組織内のメンバーアカウント間で組織レベルのダッシュボードを作成する場合には、このオプションはリージョンのみに対し使用できます。
> - 含めるか除外するバケットは最大 50 個まで選択できます。

ステップ 2: メトリクスの選択を設定する

1. [メトリクスの選択] セクションで、このダッシュボードで集約するメトリクスのタイプを選択します。

   - バケットレベルで集計され、14 日間のクエリに使用できる無料のメトリクスを含めるには、[Free metrics] (無料のメトリクス) を選択します。
   - 高度なメトリクスやその他の詳細オプションを有効にするには、[Advanced metrics and recommendations] (高度なメトリクスとレコメンデーション) を選択します。これらのオプションには、高度なプレフィックス集約、Amazon CloudWatch パブリッシング、コンテキストに応じた推奨事項が含まれます。データは 15 か月間クエリで使用できます。高度な高度な

メトリクスとレコメンデーションには追加料金がかかります。詳細については、「Amazon S3 の料金」を参照してください。

無料のメトリクスと高度なメトリクスの詳細については、「メトリクスの選択」を参照してください。

2. [Advanced metrics and recommendations features] (高度なメトリクスとレコメンデーション機能) で、有効にするオプションを選択します。

   - Advanced metrics (高度なメトリクス)
   - [CloudWatch publishing] (CloudWatch パブリッシング)
   - プレフィックス集約

   > **Important**
   >
   > S3 Storage Lens 設定で [プレフィックス集約] を有効にすると、プレフィックスレベルのメトリクスは CloudWatch に公開されません。バケット、アカウント、組織レベルの S3 Storage Lens メトリクスのみが CloudWatch に公開されます。

3. [Advanced metrics] (高度なメトリクス) を有効にした場合は、S3 ストレージレンズダッシュボードに表示する [Advanced metrics categories] (高度なメトリクスのカテゴリ) を選択します。

   - アクティビティのメトリクス
   - [Detailed status code metrics] (詳細なステータスコードメトリクス)
   - [Advanced cost optimization metrics] (高度なコスト最適化メトリクス)
   - [Advanced data protection metrics] (高度なデータ保護メトリクス)

   メトリクスのカテゴリの詳細については、「メトリクスのカテゴリ」を参照してください。メトリクスの一覧については、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

4. [プレフィックス集約] を有効にする場合は、次の設定を行います。

   a. このダッシュボードのプレフィックスしきい値の最小サイズを選択します。

      例えば、プレフィックスのしきい値を 5% とした場合は、バケットの合計ストレージサイズが 5% 以上を占めるプレフィックスが集約されることを示します。

   b. プレフィックスの深度を選択します。

この設定は、プレフィックスが評価される階層レベルの最大数を示します。プレフィックスの深度は、10 未満で指定する必要があります。

c. プレフィックスの区切り文字を入力します。

この値は、各プレフィックスのレベルを識別するために使用されます。Amazon S3 でのデフォルト値は、/ 文字ですが、実際のストレージ構造では他の区切り文字を使用することも可能です。

(オプション) ステップ 3: ダッシュボードでメトリクスをエクスポートする

1. [Metrics export] (メトリクスのエクスポート) セクションで、選択した保存先バケットに毎日出力されるメトリクスのエクスポートを作成するには、[Enable] (有効化) をクリックします。

   メトリクスのエクスポートは、CSV または Apache Parquet 形式で作成されます。メトリクスのエクスポートには、S3 Storage Lens ダッシュボードのデータと同じスコープが適用され、レコメンデーション事項は付加されません。

2. メトリクスのエクスポートを有効にした場合は、日次メトリクスのエクスポートの出力形式として、[CSV] または [Apache Parquet] を選択します。

   Parquet は、Hadoop 用のオープンソースなファイル形式で、ネストされたデータを平坦な列指向形式で格納します。

3. メトリクスのエクスポート先の S3 バケットを選択します。

   バケットは、S3 Storage Lens ダッシュボードを使用している現在のアカウント内で選択できます。また、エクスポート先バケットへの許可と、そのバケットの所有者アカウント ID がある場合は、別の AWS アカウント を選択することも可能です。

4. 送信先 S3 バケット (形式: `s3://bucket-name/prefix`) を選択します。

   バケットのアドレスは、S3 ストレージレンズダッシュボードのホームリージョンにある必要があります。S3 コンソールには、Amazon S3 によって保存先バケットのポリシーに追加される [Destination bucket permission] (エクスポート先バケットのアクセス許可) が表示されます。Amazon S3 は、エクスポート先バケットのバケットポリシーを更新し、S3 がそのバケットにデータを配置できるようにします。

5. (オプション) メトリクスのエクスポートでサーバー側の暗号化を有効にするには、[Specify an encryption key] (暗号化キーの指定) を選択します。その後、[暗号化タイプ] として [Amazon S3

マネージドキー (SSE-S3)] または [AWS Key Management Service キー (SSE-KMS)] を選択します。

キーのタイプは、Amazon S3 が管理するキー (SSE-S3) と AWS Key Management Service (AWS KMS) キー (SSE-KMS) のいずれかから選択できます。

6. (オプション) AWS KMS キーを指定するには、KMS マスターキーを選択するか、キーとなる Amazon リソースネーム (ARN) を入力する必要があります。

   カスタマーマネージドキーを選択した場合、Amazon S3 ストレージレンズに暗号化する許可を AWS KMS キーポリシーで付与する必要があります。詳細については、「AWS KMS key を使用してメトリクスのエクスポートを暗号化する」を参照してください。
7. [ダッシュボードの作成] を選択します。

ストレージをさらに可視化するには、1 つ以上の S3 Storage Lens グループを作成してダッシュボードに関連付けることができます。S3 Storage Lens グループは、プレフィックス、サフィックス、オブジェクトタグ、オブジェクトサイズ、オブジェクト経過時間、またはこれらのフィルターの組み合わせに基づいて、オブジェクト用にカスタム定義されたフィルターです。

S3 Storage Lens グループを使用すると、データレイクなどの大規模な共有バケットをきめ細かく可視化し、情報に基づいたビジネス上の意思決定を行うことができます。例えば、1 つのバケット内または複数のバケットにまたがる個々のプロジェクトやコストセンターの特定のオブジェクトグループにストレージ使用量を分類することで、ストレージ割り当てを効率化し、コストレポートを最適化できます。

S3 Storage Lens グループを使用するには、高度なメトリクスとレコメンデーションを使用するようにダッシュボードをアップグレードする必要があります。S3 Storage Lens グループの詳細については、「the section called “S3 Storage Lens グループの使用”」を参照してください。

Amazon S3 Storage Lens ダッシュボードの更新

Amazon S3 コンソールで Amazon S3 Storage Lens ダッシュボードを更新するには、次の手順に従います。

ステップ 1: ダッシュボードのスコープを更新する

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。

3. 編集するダッシュボードを選択し、 [編集] をクリックします。

   Edit dashboard (ダッシュボードの編集) ページが開きます。

   > **Note**
   >
   > 次の項目は変更できません。
   >
   > - ダッシュボード名
   > - ホームリージョン
   > - デフォルトのダッシュボードの (アカウントのストレージ全体を対象としている) ダッシュボードスコープ

4. ダッシュボード設定ページの [General] (全般) セクションで、更新およびダッシュボードにタグを追加します。

   タグを使用して、ダッシュボードへのアクセス許可を管理し、また S3 Storage Lens のコストを追跡できます。詳細については、IAM ユーザーガイドのリソースタグを使用したアクセスの制御、および AWS Billing ユーザーガイドの AWS 生成コスト配分タグを参照してください。

   > **Note**
   >
   > ダッシュボードの設定には、最大 50 個のタグを追加できます。

5. [ダッシュボードスコープ] セクションで、次の操作を行います。

   a. S3 Storage Lens がダッシュボードに含める、または除外するリージョンとバケットを更新します。

      > **Note**
      >
      > - リージョンとバケットを、含めるか除外するかの選択が可能です。組織内のメンバーアカウント間で組織レベルのダッシュボードを作成する場合には、このオプションはリージョンのみに対し使用できます。
      > - 含めるか除外するバケットは最大 50 個まで選択できます。

   b. 選択したリージョンにあるバケットを更新し、S3 Storage Lens に含めるか、除外するかを指定します。バケットを含めるか除外するかの設定は可能ですが、両方を同時に設定するこ

とはできません。このオプションは、組織レベルのダッシュボードを作成している場合には表示されません。

ステップ 2: メトリクスの選択を更新する

1. [メトリクスの選択] セクションで、このダッシュボードで集約するメトリクスのタイプを選択します。

   - バケットレベルで集計され、14 日間のクエリに使用できる無料のメトリクスを含めるには、[Free metrics] (無料のメトリクス) を選択します。
   - 高度なメトリクスやその他の詳細オプションを有効にするには、[Advanced metrics and recommendations] (高度なメトリクスとレコメンデーション) を選択します。これらのオプションには、高度なプレフィックス集約、Amazon CloudWatch パブリッシング、コンテキストに応じた推奨事項が含まれます。データは 15 か月間クエリで使用できます。高度な高度なメトリクスとレコメンデーションには追加料金がかかります。詳細については、「Amazon S3 の料金」を参照してください。

     無料のメトリクスと高度なメトリクスの詳細については、「メトリクスの選択」を参照してください。

2. [Advanced metrics and recommendations features] (高度なメトリクスとレコメンデーション機能) で、有効にするオプションを選択します。

   - Advanced metrics (高度なメトリクス)
   - [CloudWatch publishing] (CloudWatch パブリッシング)
   - プレフィックス集約

   > **Important**
   >
   > S3 Storage Lens 設定で [プレフィックス集約] を有効にすると、プレフィックスレベルのメトリクスは CloudWatch に公開されません。バケット、アカウント、組織レベルの S3 Storage Lens メトリクスのみが CloudWatch に公開されます。

3. [Advanced metrics] (高度なメトリクス) を有効にした場合は、S3 ストレージレンズダッシュボードに表示する [Advanced metrics categories] (高度なメトリクスのカテゴリ) を選択します。

   - アクティビティのメトリクス
   - [Detailed status code metrics] (詳細なステータスコードメトリクス)

- [Advanced cost optimization metrics] (高度なコスト最適化メトリクス)
- [Advanced data protection metrics] (高度なデータ保護メトリクス)

メトリクスカテゴリの詳細については、「メトリクスのカテゴリ」を参照してください。メトリクスの一覧については、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

4. [プレフィックス集約] を有効にする場合は、次の設定を行います。

   a. このダッシュボードのプレフィックスしきい値の最小サイズを選択します。

      例えば、プレフィックスのしきい値を 5% とした場合は、バケットの合計ストレージサイズが 5% 以上を占めるプレフィックスが集約されることを示します。

   b. プレフィックスの深度を選択します。

      この設定は、プレフィックスが評価される階層レベルの最大数を示します。プレフィックスの深度は、10 未満で指定する必要があります。

   c. プレフィックスの区切り文字を入力します。

      これは、各プレフィックスのレベルを識別するために使用される値です。Amazon S3 でのデフォルト値は、/ 文字ですが、実際のストレージ構造では他の区切り文字を使用することも可能です。

(オプション) ステップ 3: ダッシュボードでメトリクスをエクスポートする

1. [Metrics export] (メトリクスのエクスポート) セクションで、選択した保存先バケットに毎日出力されるメトリクスのエクスポートを作成するには、[Enable] (有効化) をクリックします。メトリクスのエクスポートを無効にするには、[Disable] (無効化) を選択します。

   メトリクスのエクスポートは、CSV または Apache Parquet 形式で作成されます。メトリクスのエクスポートには、S3 Storage Lens ダッシュボードのデータと同じスコープが適用され、レコメンデーション事項は付加されません。

2. この機能を有効にした場合は、日次メトリクスのエクスポートの出力形式として、[CSV] または [Apache Parquet] を選択します。

   Parquet は、Hadoop 用のオープンソースなファイル形式で、ネストされたデータを平坦な列指向形式で格納します。

3. メトリクスのエクスポート先の S3 バケットを選択します。

バケットは、S3 Storage Lens ダッシュボードを使用している現在のアカウント内で選択できます。また、エクスポート先バケットへの許可と、そのバケットの所有者アカウント ID がある場合は、別の AWS アカウント を選択することも可能です。

4. 送信先 S3 バケット (形式: `s3://bucket-name/prefix`) を選択します。

   バケットのアドレスは、S3 ストレージレンズダッシュボードのホームリージョンにある必要があります。S3 コンソールには、Amazon S3 によって保存先バケットのポリシーに追加される [Destination bucket permission] (エクスポート先バケットのアクセス許可) が表示されます。Amazon S3 は、エクスポート先バケットのバケットポリシーを更新し、S3 がそのバケットにデータを配置できるようにします。

5. (オプション) メトリクスのエクスポートでサーバー側の暗号化を有効にするには、[Specify an encryption key] (暗号化キーの指定) を選択します。その後、[暗号化タイプ] として [Amazon S3 マネージドキー (SSE-S3)] または [AWS Key Management Service キー (SSE-KMS)] を選択します。

   キーのタイプは、Amazon S3 が管理するキー (SSE-S3) と AWS Key Management Service (AWS KMS) キー (SSE-KMS) のいずれかから選択できます。

6. (オプション) AWS KMS キーを指定するには、KMS マスターキーを選択するか、キーとなる Amazon リソースネーム (ARN) を入力する必要があります。[AWS KMS キー] で、次のいずれかの方法で KMS キーを指定します。

   - 使用可能な KMS キーのリストから選択するには、[AWS KMS keys から選択する] を選択し、使用可能なキーのリストから自分の KMS キーを選択します。

     AWS マネージドキー (aws/s3) とカスタマーマネージドキーの両方がこのリストに表示されます。カスタマーマネージドキーの詳細については、AWS Key Management Serviceデベロッパーガイドの「カスタマーキーと AWS キー」を参照してください。

     > **Note**
     >
     > AWS マネージドキー (aws/s3) は S3 ストレージレンズでの SSE-KMS 暗号化ではサポートされていません。

   - KMS キー ARN を入力するには、[AWS KMS key ARN を入力] を選択し、表示されるフィールドに KMS キー ARN を入力します。
   - AWS KMS コンソールで新しいカスタマーマネージドキーを作成するには、[KMS キーを作成] を選択します。

カスタマーマネージドキーを選択した場合、Amazon S3 ストレージレンズに暗号化する許可を AWS KMS キーポリシーで付与する必要があります。詳細については、「AWS KMS key を使用してメトリクスのエクスポートを暗号化する」を参照してください。

AWS KMS key の作成の詳細については、AWS Key Management Service デベロッパーガイドのキーの作成を参照してください。

7. [Save changes] (変更の保存) をクリックします。

ストレージをさらに可視化するには、1 つ以上の S3 Storage Lens グループを作成してダッシュボードに関連付けることができます。S3 Storage Lens グループは、プレフィックス、サフィックス、オブジェクトタグ、オブジェクトサイズ、オブジェクト経過時間、またはこれらのフィルターの組み合わせに基づいて、オブジェクト用にカスタム定義されたフィルターです。

S3 Storage Lens グループを使用すると、データレイクなどの大規模な共有バケットをきめ細かく可視化し、情報に基づいたビジネス上の意思決定を行うことができます。例えば、1 つのバケット内または複数のバケットにまたがる個々のプロジェクトやコストセンターの特定のオブジェクトグループにストレージ使用量を分類することで、ストレージ割り当てを効率化し、コストレポートを最適化できます。

S3 Storage Lens グループを使用するには、高度なメトリクスとレコメンデーションを使用するようにダッシュボードをアップグレードする必要があります。S3 Storage Lens グループの詳細については、「the section called “S3 Storage Lens グループの使用”」を参照してください。

Amazon S3 Storage Lens ダッシュボードの無効化と削除

S3 ストレージレンズはメトリクスを集約し、Amazon S3 コンソールの [Buckets] (バケット) ページの [Account snapshot] (アカウントスナップショット) セクションにこの情報を表示します。S3 Storage Lens は、インサイトと傾向を可視化したり、外れ値にフラグ付けしたり、ストレージコストの最適化やデータ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取ったりするために使用できるインタラクティブダッシュボードも提供します。ダッシュボードには、組織、アカウント、AWS リージョン、バケット、オブジェクト、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成して可視化できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。

Amazon S3 Storage Lens のデフォルトのダッシュボードは、default-account-dashboard です。このダッシュボードは Amazon S3 によって事前定義されており、無料のメトリクスおよび高度なメトリ

クスをアカウント全体で集計し、その概要と傾向をコンソール上で可視化するのに役立ちます。デフォルトダッシュボードのスコープ設定を変更することはできませんが、メトリクスの選択を無料メトリクスから有料の高度なメトリクスとレコメンデーションにアップグレードしたり、オプションのメトリクスエクスポートを構成したり、デフォルトダッシュボードを無効にすることは可能です。デフォルトのダッシュボードは削除できません。

Amazon S3 Storage Lens ダッシュボードは、Amazon S3 コンソールから削除または無効にすることができます。無効もしくは削除されたダッシュボードからは、その後メトリクスは生成されなくなります。無効化されたダッシュボードは引き続き設定情報を保持するため、簡単に再有効化して復帰させることができます。無効化されたダッシュボードは、クエリに使用できなくなるまで、履歴データを保持します。

無料メトリクス選択のデータは、クエリのために 14 日間利用でき、高度なメトリクスとレコメンデーションの選択のためのデータはクエリのために15 か月間利用できます。

Amazon S3 Storage Lens ダッシュボードの無効化

S3 Storage Lens ダッシュボードを無効にするには、

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [ダッシュボード] の一覧から、無効にするダッシュボードを選択し、その一覧の上部にある [無効化] をクリックします。
4. 確認ページで、ダッシュボードの無効化を確認するために、テキストフィールドに対象のダッシュボードの名前を入力した上で、[確認] をクリックします。

Amazon S3 Storage Lens ダッシュボードの削除

> **Note**
>
> デフォルトのダッシュボードを削除することはできません。ただし、無効にすることは可能です。作成したダッシュボードを削除する前に、次の点を考慮してください。
>
> - ダッシュボードを削除する代わりに、無効化することもできます。この場合、将来的に同じダッシュボードを再度有効化できます。詳細については、「Amazon S3 Storage Lens ダッシュボードの無効化」を参照してください。

- ダッシュボードを削除すると、そのダッシュボードに関連付けられているすべての構成設定が削除されます。
- また、ダッシュボードを削除すると、すべての履歴メトリクスデータも利用できなくなります。この履歴データは 15 か月間保持されます。このデータに再度アクセスする場合は、削除されたものと同じホームリージョンで、同じ名前のダッシュボードを再度作成します。

S3 Storage Lens ダッシュボードを削除するには、

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Dashboards] (ダッシュボード) の順にクリックします。
3. [ダッシュボード] の一覧から、削除するダッシュボードを選択した上で、その一覧の上部にある [削除] をクリックします。
4. ダッシュボードの削除を確認するために、ダッシュボードの削除ページのテキストフィールドに、対象のダッシュボードの名前を入力します。[Confirm] (確認) を選択します。

組織レベルのダッシュボードを作成するための AWS Organizations の使用

S3 ストレージレンズはメトリクスを集約し、Amazon S3 コンソールの [Buckets] (バケット) ページの [Account snapshot] (アカウントスナップショット) セクションにこの情報を表示します。S3 Storage Lens は、インサイトと傾向を可視化したり、外れ値にフラグ付けしたり、ストレージコストの最適化やデータ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取ったりするために使用できるインタラクティブダッシュボードも提供します。ダッシュボードには、組織、アカウント、AWS リージョン、バケット、オブジェクト、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成して可視化できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。

Amazon S3 Storage Lens のデフォルトのダッシュボードは、default-account-dashboard です。このダッシュボードは Amazon S3 によって事前定義されており、無料のメトリクスおよび高度なメトリクスをアカウント全体で集計し、その概要と傾向をコンソール上で可視化するのに役立ちます。デフォルトダッシュボードのスコープ設定を変更することはできませんが、メトリクスの選択を無料メトリクスから有料の高度なメトリクスとレコメンデーションにアップグレードしたり、オプションの

メトリクスエクスポートを構成したり、デフォルトダッシュボードを無効にすることは可能です。デフォルトのダッシュボードは削除できません。

また、組織内の特定の AWS リージョン、S3 バケット、他の AWS アカウントなどに対応するための、追加の S3 Storage Lens ダッシュボードを作成することもできます。

S3 Storage Lens ダッシュボードには、ストレージスコープに関する豊富な情報リソースが表示されます。ダッシュボードは 30 を超えるメトリクスを可視化し、これらのメトリクスからは、傾向や情報 (ストレージの概要、コスト効率、データ保護、アクティビティなど) が得られます。

Amazon S3 Storage Lens を使用することで、AWS Organizations の階層に属しているすべてのアカウントから、ストレージのメトリクスと使用状況に関するデータを収集できます。これを行うには、AWS Organizations を使用している必要があります。また、AWS Organizations 管理アカウントを使用して、S3 ストレージレンズに対し、信頼されたアクセスを有効にする必要があります。

信頼されたアクセスを有効にすると、組織内のアカウントに対し、委任された管理者のアクセス権限を追加できます。これらのアカウントでは、S3 Storage Lens のダッシュボードと設定を、組織の全体に対して作成できます。信頼されたアクセスの有効化の詳細については、AWS Organizations ユーザーガイドの Amazon S3 Lens および AWS Organizations を参照してください。

次に示すコンソールコントロールは、AWS Organizations の管理アカウントでのみ使用が可能です。

組織内の S3 Storage Lens に対する、信頼されたアクセスの有効化

信頼されたアクセスが有効化された Amazon S3 ストレージレンズでは、AWS Organizations API オペレーションを通じて、AWS Organizations の階層やメンバーシップ、および構造にアクセスできるようになります。S3 ストレージレンズは、組織全体の構造で信頼されたサービスとして認識されます。ダッシュボードの設定が作成されるたびに、組織内の管理アカウントまたは委任された管理者アカウントのために、サービスにリンクされたロールを作成できます。

サービスにリンクされたロールからは S3 Storage Lens に対し、組織の設定、アカウントの一覧作成、組織のためのサービスアクセスに関する一覧の確認、組織のための委任された管理者の取得などを行う権限が付与されます。これにより S3 ストレージレンズは、組織内のアカウントのダッシュボードのために、ストレージの使用状況とアクティビティに関する、クロスアカウントのメトリクスを収集できるようになります。

詳細については、「Amazon S3 ストレージレンズでのサービスにリンクされたロールの使用」を参照してください。

> **Note**
>
> - 信頼されたアクセスは、管理アカウントからのみ有効化できます。
> - 組織に対し、S3 Storage Lens ダッシュボードまたは、その設定を作成できるのは、管理アカウントと委任された管理者のみです。

S3 Storage Lens で信頼されたアクセスを有効にするには、

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Organization settings] (組織の設定) の順にクリックします。
3. [組織のアクセス] で、[編集] をクリックします。

   組織のアクセスページが開きます。このページで、S3 Storage Lens のために、信頼されたアクセスを有効化できます。これを有効化したユーザーと、そのユーザーにより委任された管理者として追加された他のアカウント所有者は、組織内のすべてのアカウントとストレージ用に、ダッシュボードを作成できるようになります。

組織内で S3 Storage Lens の信頼されたアクセスを無効にする

信頼されたアクセスを無効にすると、S3 ストレージレンズの機能が、アカウントレベルのみに制限されます。各アカウント保有者は、自身のアカウントの範囲に限定して S3 Storage Lens が提供する機能にアクセスでき、組織レベルではアクセスできなくなります。信頼されたアクセスを必要とするダッシュボードは更新されなくなりますが、それぞれのデータがクエリで利用可能な期間中はダッシュボードで履歴データをクエリすることができます。

委任された管理者としてのアカウントを削除すると、アカウント所有者の S3 ストレージレンズのダッシュボードによるメトリクスへのアクセスは、アカウントレベルでのみ機能するように制限されます。それまでに作成した、組織レベルのすべてのダッシュボードはもはや更新されなくなります。ただし [the period that it is available for queries] ごとの履歴データはクエリが実行できるようになります。

> **Note**
>
> - 信頼されたアクセスを無効にすると、組織レベルのダッシュボードもすべて自動的に無効になります。ストレージについてのメトリクスを収集および集計するための、組織アカウントへの信頼されたアクセス権が、S3 Storage Lens からなくなるためです。
> - これらの無効化されたダッシュボードの履歴アカウントはこれらの無効化されたダッシュボードのためのデータを以前見ることができ、また使用可能であればこのデータをクエリできます。

S3 Storage Lens の信頼されたアクセスを無効にするには、

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Organization settings] (組織の設定) の順にクリックします。
3. [組織のアクセス] で、[編集] をクリックします。

   組織のアクセスページが開きます。このページで、S3 Storage Lens の信頼されたアクセスを無効化できます。

S3 Storage Lens のための委任された管理者の登録

信頼されたアクセスを有効にした後で、組織内のアカウントに対する委任された管理者のアクセス権限を登録できます。アカウントが委任された管理者として登録されると、そのアカウントがら AWS Organizations のすべての読み取り専用 API オペレーションへアクセスすることが承認されます。これにより、ユーザーに代わって S3 Storage Lens ダッシュボードを作成するための可視性が、組織のメンバーと構造に対し提供されます。

S3 Storage Lens のための委任された管理者を登録するには、

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Organization settings] (組織の設定) の順にクリックします。
3. [委任されたアクセス] セクションの [アカウント] で、[アカウントの追加] をクリックします。

委任された管理者のアクセスページが開きます。このページでは、AWS アカウント ID を委任管理者として追加し、組織内のすべてのアカウントとストレージのために組織レベルのダッシュボードを作成することを許可できます。

S3 Storage Lens の委任された管理者の登録解除

組織内のアカウントのための、委任された管理者としてのアクセス権限を、登録解除することができます。アカウントの委任された管理者としての登録が解除されると、そのアカウントは、AWS Organizations の読み取り専用 API オペレーションすべてに対するアクセス権限を失い、組織のメンバーや構造にアクセスできなくなります。

> **Note**
>
> - 委任された管理者の登録が解除されると、委任された管理者としてそれまでに作成した組織レベルのダッシュボードも、すべて自動的に無効になります。
> - 委任された管理者アカウントは、データがクエリのために利用できるそれぞれの保持期間に従い、これらの無効化されたダッシュボードの履歴データを、引き続き見ることが可能です。

アカウントでの、委任された管理者としてのアクセス権限の登録を解除するには、

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、[Storage Lens] (ストレージレンズ)、[Organization settings] (組織の設定) の順にクリックします。
3. [アクセスが委任されたアカウント] セクションで、登録を解除するアカウント ID を選択した上で、[削除] をクリックします。

AWS CLI を使用した Amazon S3 ストレージレンズの例

S3 ストレージレンズはメトリクスを集約し、Amazon S3 コンソールの [Buckets] (バケット) ページの [Account snapshot] (アカウントスナップショット) セクションにこの情報を表示します。S3 Storage Lens は、インサイトと傾向を可視化したり、外れ値にフラグ付けしたり、ストレージコストの最適化やデータ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取るために使用できるインタラクティブダッシュボードも提供します。ダッシュボードには、組織、アカ

ウント、AWS リージョン、バケット、オブジェクト、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成して可視化できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。詳細については、「Amazon S3 ストレージレンズを使用してのストレージアクティビティと使用状況の評価」を参照してください。

次の例は、AWS Command Line Interface を通じて S3 ストレージレンズを使用する方法を示しています。

トピック

- Amazon S3 ストレージレンズを使用するためのヘルパーファイル
- AWS CLI による Amazon S3 ストレージレンズ設定の使用
- AWS CLI を使用した AWS Organizations の例での Amazon S3 ストレージレンズの使用

Amazon S3 ストレージレンズを使用するためのヘルパーファイル

次の JSON ファイルとキー入力を例として使用します。

JSON の S3 Storage Lens 例の設定

Example **config.json**

config.json ファイルには、S3 ストレージレンズの組織レベルの高度なメトリクスとレコメンデーションに関する設定の詳細が含まれています。次の例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

> **Note**
>
> 高度なメトリクスとレコメンデーションには追加料金が適用されます。詳細については、「高度なメトリクスとレコメンデーション事項」を参照してください。

```
{
  "Id": "SampleS3StorageLensConfiguration", //Use this property to identify your S3
 Storage Lens configuration.
  "AwsOrg": { //Use this property when enabling S3 Storage Lens for AWS Organizations.
    "Arn": "arn:aws:organizations::123456789012:organization/o-abcdefgh"
  },
  "AccountLevel": {
    "ActivityMetrics": {
```

```
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "DetailedStatusCodesMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
        "IsEnabled":true
      },
      "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
      },
      "AdvancedCostOptimizationMetrics": {
        "IsEnabled":true
      },
      "DetailedStatusCodesMetrics": {
        "IsEnabled":true
      },
      "PrefixLevel":{
        "StorageMetrics":{
          "IsEnabled":true,
          "SelectionCriteria":{
            "MaxDepth":5,
            "MinStorageBytesPercentage":1.25,
            "Delimiter":"/"
          }
        }
      }
    }
  },
  "Exclude": { //Replace with "Include" if you prefer to include Regions.
    "Regions": [
      "eu-west-1"
    ],
    "Buckets": [ //This attribute is not supported for AWS Organizations-level
configurations.
      "arn:aws:s3:::source_bucket1"
    ]
```

```
  },
  "IsEnabled": true, //Whether the configuration is enabled
  "DataExport": { //Details about the metrics export
    "S3BucketDestination": {
      "OutputSchemaVersion": "V_1",
      "Format": "CSV", //You can add "Parquet" if you prefer.
      "AccountId": "111122223333",
      "Arn": "arn:aws:s3:::destination-bucket-name", // The destination bucket for your
 metrics export must be in the same Region as your S3 Storage Lens configuration.
      "Prefix": "prefix-for-your-export-destination",
      "Encryption": {
        "SSES3": {}
      }
    },
    "CloudWatchMetrics": {
      "IsEnabled": true
    }
  }
}
```

JSON の Storage Lens グループを使用した S3 Storage Lens の設定例

Example **`config.json`**

`config.json` ファイルには、Storage Lens グループを使用する際に Storage Lens 設定に適用する詳細が含まれています。例を使用するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

すべての Storage Lens グループをダッシュボードにアタッチするには、Storage Lens 設定を以下の構文で更新します。

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
```

```
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
    "StorageLensGroupLevel": {},
  "IsEnabled": true
}
```

Storage Lens ダッシュボード設定に 2 つの Storage Lens グループ (*slg-1* と *slg-2*) のみを含めるには、次の構文を使用します。

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
   "StorageLensGroupLevel": {
        "SelectionCriteria": {
            "Include": [
                "arn:aws:s3:us-east-1:111122223333:storage-lens-group/slg-1",
                "arn:aws:s3:us-east-1:444455556666:storage-lens-group/slg-2"
            ]
    },
  "IsEnabled": true
}
```

特定の Storage Lens グループのみがダッシュボード設定にアタッチされないようにするには、次の構文を使用します。

```
{
  "Id": "ExampleS3StorageLensConfiguration",
```

```
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
   "StorageLensGroupLevel": {
        "SelectionCriteria": {
            "Exclude": [
                "arn:aws:s3:us-east-1:111122223333:storage-lens-group/slg-1",
                "arn:aws:s3:us-east-1:444455556666:storage-lens-group/slg-2"
            ]
    },
  "IsEnabled": true
}
```

JSON の S3 Storage Lens タグ例の設定

Example **tags.json**

tags.json ファイルには、S3 ストレージレンズ設定に適用するためのタグが含まれています。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
[
    {
        "Key": "key1",
        "Value": "value1"
    },
    {
        "Key": "key2",
        "Value": "value2"
    }
]
```

S3 Storage Lens での IAM アクセス許可の設定例

Example **permissions.json** — 特定のダッシュボード名

このポリシー例は、特定のダッシュボード名が指定されている S3 ストレージレンズ IAM アクセス許可を示しています。*value1*、*us-east-1*、*your-dashboard-name*、*example-account-id* を独自の値に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetStorageLensConfiguration",
                "s3:DeleteStorageLensConfiguration",
                "s3:PutStorageLensConfiguration"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/key1": "value1"
                }
            },
            "Resource": "arn:aws:s3:us-east-1:example-account-id:storage-lens/your-
dashboard-name"
        }
    ]
}
```

Example **permissions.json** — 特定のダッシュボード名がない場合

このポリシー例は、特定のダッシュボード名が指定されていない S3 ストレージレンズ IAM アクセス許可を示しています。*value1*、*us-east-1*、*example-account-id* を独自の値に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetStorageLensConfiguration",
                "s3:DeleteStorageLensConfiguration",
                "s3:PutStorageLensConfiguration"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/key1": "value1"
                }
            },
            "Resource": "arn:aws:s3:us-east-1:example-account-id:storage-lens/*"
        }
    ]
}
```

AWS CLI による Amazon S3 ストレージレンズ設定の使用

AWS CLI を使用して、S3 ストレージレンズ設定の一覧表示、作成、削除、取得、タグ付け、更新を行うことができます。次の例では、キーの入力用にヘルパー JSON ファイルを使用しています。これらの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

S3 ストレージレンズ設定を作成する

Example S3 ストレージレンズ設定を作成する

```
aws s3control put-storage-lens-configuration --account-id=111122223333 --
config-id=example-dashboard-configuration-id --region=us-east-1 --storage-lens-
configuration=file://./config.json --tags=file://./tags.json
```

タグなしで S3 ストレージレンズ設定を作成する

Example タグなしで S3 ストレージレンズ設定を作成する

```
aws s3control put-storage-lens-configuration --account-id=222222222222 --config-
id=your-configuration-id --region=us-east-1 --storage-lens-configuration=file://./
config.json
```

S3 Storage Lens 設定を取得する

Example S3 Storage Lens 設定を取得する

```
aws s3control get-storage-lens-configuration --account-id=222222222222 --config-
id=your-configuration-id --region=us-east-1
```

続きのトークンなしで S3 ストレージレンズ設定を一覧表示する

Example 続きのトークンなしで S3 ストレージレンズ設定を一覧表示する

```
aws s3control list-storage-lens-configurations --account-id=222222222222 --region=us-east-1
```

S3 Storage Lens 設定を一覧表示する

Example S3 Storage Lens 設定を一覧表示する

```
aws s3control list-storage-lens-configurations --account-id=222222222222 --region=us-east-1 --next-token=abcdefghij1234
```

S3 Storage Lens 設定を削除する

Example S3 Storage Lens 設定を削除する

```
aws s3control delete-storage-lens-configuration --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

S3 ストレージレンズ設定にタグを追加する

Example S3 ストレージレンズ設定にタグを追加する

```
aws s3control put-storage-lens-configuration-tagging --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id --tags=file://./tags.json
```

S3 Storage Lens 設定のタグを取得する

Example S3 Storage Lens 設定のタグを取得する

```
aws s3control get-storage-lens-configuration-tagging --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

S3 Storage Lens 設定のタグを削除する

Example S3 Storage Lens 設定のタグを削除する

```
aws s3control delete-storage-lens-configuration-tagging --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

AWS CLI を使用した AWS Organizations の例での Amazon S3 ストレージレンズの使用

Amazon S3 ストレージレンズを使用して、AWS Organizations の階層に属しているすべてのアカウントから、ストレージのメトリクスと使用状況に関するデータを収集できます。詳細については、「AWS Organizations とともに Amazon S3 ストレージレンズを使用する」を参照してください。

S3 Storage Lens のために組織の信頼されたアクセスを有効にする

Example S3 Storage Lens のために組織の信頼されたアクセスを有効にする

```
aws organizations enable-aws-service-access --service-principal storage-
lens.s3.amazonaws.com
```

S3 Storage Lens で組織の信頼されたアクセスを無効にする

Example S3 Storage Lens で組織の信頼されたアクセスを無効にする

```
aws organizations disable-aws-service-access --service-principal storage-
lens.s3.amazonaws.com
```

S3 Storage Lens のために組織の委任された管理者を登録する

Example S3 Storage Lens のために組織の委任された管理者を登録する

この例を使用するには、*111122223333* を適切な AWS アカウント ID で置き換えてください。

```
aws organizations register-delegated-administrator --service-principal storage-
lens.s3.amazonaws.com --account-id 111122223333
```

S3 Storage Lens で委任された Organizations の管理者の登録を解除する

Example S3 Storage Lens で委任された Organizations の管理者の登録を解除する

この例を使用するには、*111122223333* を適切な AWS アカウント ID で置き換えてください。

```
aws organizations deregister-delegated-administrator --service-principal storage-
lens.s3.amazonaws.com --account-id 111122223333
```

SDK for Java を使用した Amazon S3 ストレージレンズの例

S3 ストレージレンズはメトリクスを集約し、Amazon S3 コンソールの [Buckets] (バケット) ページの [Account snapshot] (アカウントスナップショット) セクションにこの情報を表示します。S3

Storage Lens は、インサイトと傾向を可視化したり、外れ値にフラグ付けしたり、ストレージコストの最適化やデータ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取るために使用できるインタラクティブダッシュボードも提供します。ダッシュボードには、組織、アカウント、AWS リージョン、バケット、オブジェクト、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成して可視化できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。詳細については、「Amazon S3 ストレージレンズを使用してのストレージアクティビティと使用状況の評価」を参照してください。

次の例は、AWS SDK for Java を通じて S3 ストレージレンズを使用する方法を示しています。

トピック

- SDK for Java を使用して Amazon S3 ストレージレンズ設定を使用する

SDK for Java を使用して Amazon S3 ストレージレンズ設定を使用する

S3 Storage Lens 設定の一覧表示、作成、取得、更新を、SDK for Java から行うことができます。次の例では、キーの入力用にヘルパー JSON ファイルを使用しています。

トピック

- S3 Storage Lens 設定を作成および更新する
- S3 Storage Lens 設定を削除する
- S3 Storage Lens 設定を取得する
- S3 Storage Lens 設定を一覧表示する
- S3 ストレージレンズ設定にタグを追加する
- S3 Storage Lens 設定のタグを取得する
- S3 Storage Lens 設定のタグを削除する
- 高度な推奨のメトリクスを使用して S3 ストレージレンズのデフォルト設定を更新する
- Storage Lens のグループを S3 Storage Lens のダッシュボードにアタッチする
- SDK for Java を使用した AWS Organizations の例で Amazon S3 ストレージレンズを使用する

S3 Storage Lens 設定を作成および更新する

Example S3 Storage Lens 設定を作成および更新する

```
package aws.example.s3control;
```

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.CloudWatchMetrics;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateAndUpdateDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        String exportAccountId = "Destination Account ID";
        String exportBucketArn = "arn:aws:s3:::destBucketName"; // The destination
 bucket for your metrics export must be in the same Region as your S3 Storage Lens
 configuration.
        String awsOrgARN = "arn:aws:organizations::123456789012:organization/o-
abcdefgh";
        Format exportFormat = Format.CSV;

        try {
```

```
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            Include include = new Include()
                    .withBuckets(Arrays.asList("arn:aws:s3:::bucketName"))
                    .withRegions(Arrays.asList("us-west-2"));

            StorageLensDataExportEncryption exportEncryption = new
StorageLensDataExportEncryption()
                    .withSSES3(new SSES3());
            S3BucketDestination s3BucketDestination = new S3BucketDestination()
                    .withAccountId(exportAccountId)
                    .withArn(exportBucketArn)
                    .withEncryption(exportEncryption)
                    .withFormat(exportFormat)
                    .withOutputSchemaVersion(OutputSchemaVersion.V_1)
                    .withPrefix("Prefix");
            CloudWatchMetrics cloudWatchMetrics = new CloudWatchMetrics()
                    .withIsEnabled(true);
```

```
            StorageLensDataExport dataExport = new StorageLensDataExport()
                    .withCloudWatchMetrics(cloudWatchMetrics)
                    .withS3BucketDestination(s3BucketDestination);

            StorageLensAwsOrg awsOrg = new StorageLensAwsOrg()
                    .withArn(awsOrgARN);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withInclude(include)
                    .withDataExport(dataExport)
                    .withAwsOrg(awsOrg)
                    .withIsEnabled(true);

            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

S3 Storage Lens 設定を削除する

Example S3 Storage Lens 設定を削除する

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class DeleteDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.deleteStorageLensConfiguration(new
 DeleteStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
```

```
}
```

S3 Storage Lens 設定を取得する

Example S3 Storage Lens 設定を取得する

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.GetStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.GetStorageLensConfigurationResult;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class GetDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final StorageLensConfiguration configuration =
                    s3ControlClient.getStorageLensConfiguration(new
 GetStorageLensConfigurationRequest()
                            .withAccountId(sourceAccountId)
                            .withConfigId(configurationId)
                    ).getStorageLensConfiguration();

            System.out.println(configuration.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
```

```
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

S3 Storage Lens 設定を一覧表示する

Example S3 Storage Lens 設定を一覧表示する

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.ListStorageLensConfigurationEntry;
import com.amazonaws.services.s3control.model.ListStorageLensConfigurationsRequest;

import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class ListDashboard {

    public static void main(String[] args) {
        String sourceAccountId = "Source Account ID";
        String nextToken = "nextToken";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final List<ListStorageLensConfigurationEntry> configurations =
                    s3ControlClient.listStorageLensConfigurations(new
 ListStorageLensConfigurationsRequest()
                            .withAccountId(sourceAccountId)
                            .withNextToken(nextToken)
                    ).getStorageLensConfigurationList();
```

```
            System.out.println(configurations.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

S3 ストレージレンズ設定にタグを追加する

Example S3 ストレージレンズ設定にタグを追加する

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import
 com.amazonaws.services.s3control.model.PutStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class PutDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";

        try {
            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );
```

```
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfigurationTagging(new
 PutStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

S3 Storage Lens 設定のタグを取得する

Example S3 Storage Lens 設定のタグを取得する

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationRequest;
import
 com.amazonaws.services.s3control.model.GetStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;
```

```
public class GetDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final List<StorageLensTag> s3Tags = s3ControlClient
                    .getStorageLensConfigurationTagging(new
 GetStorageLensConfigurationTaggingRequest()
                            .withAccountId(sourceAccountId)
                            .withConfigId(configurationId)
                    ).getTags();

            System.out.println(s3Tags.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

S3 Storage Lens 設定のタグを削除する

Example S3 Storage Lens 設定のタグを削除する

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import
 com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationTaggingRequest;
```

```
import static com.amazonaws.regions.Regions.US_WEST_2;

public class DeleteDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.deleteStorageLensConfigurationTagging(new
 DeleteStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

高度な推奨のメトリクスを使用して S3 ストレージレンズのデフォルト設定を更新する

Example 高度なメトリクスとレコメンデーション事項を使用して S3 ストレージレンズのデフォルト設定を更新する

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
```

```
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateDefaultConfigWithPaidFeatures {

    public static void main(String[] args) {
        String configurationId = "default-account-dashboard"; // This configuration ID
 cannot be modified.
        String sourceAccountId = "Source Account ID";

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
 PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
```

```
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

> **Note**
>
> 高度なメトリクスとレコメンデーション事項には追加料金が適用されます。詳細については、「高度なメトリクスとレコメンデーション事項」を参照してください。

Storage Lens のグループを S3 Storage Lens のダッシュボードにアタッチする

Example すべての Storage Lens グループをダッシュボードにアタッチします。

次の SDK for Java の例では、アカウント *111122223333* のすべての Storage Lens グループを *DashboardConfigurationId* ダッシュボードにアタッチします。

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWithStorageLensGroups {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel();

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();
```

```
            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example 2 つの Storage Lens グループをダッシュボードにアタッチする

次の AWS SDK for Java 例では、2 つの Storage Lens グループ (*StorageLensGroupName1* と *StorageLensGroupName2*) を *ExampleDashboardConfigurationId* ダッシュボードにアタッチします。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;
import com.amazonaws.services.s3control.model.StorageLensGroupLevelSelectionCriteria;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWith2StorageLensGroups {
    public static void main(String[] args) {
```

```
        String configurationId = "ExampleDashboardConfigurationId";
        String storageLensGroupName1 = "StorageLensGroupName1";
        String storageLensGroupName2 = "StorageLensGroupName2";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevelSelectionCriteria selectionCriteria = new
StorageLensGroupLevelSelectionCriteria()
                    .withInclude(
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
+ ":storage-lens-group/" + storageLensGroupName1,
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
+ ":storage-lens-group/" + storageLensGroupName2);

            System.out.println(selectionCriteria);
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel()
                    .withSelectionCriteria(selectionCriteria);

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
```

```
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example 除外項目を含むすべての Storage Lens グループをアタッチする

次の SDK for Java 例では、2 つの Storage Lens グループを *ExampleDashboardConfigurationId* ダッシュボードにアタッチします (指定した *StorageLensGroupName1* と *StorageLensGroupName2* を除く):

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;
import com.amazonaws.services.s3control.model.StorageLensGroupLevelSelectionCriteria;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWith2StorageLensGroupsExcluded {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String storageLensGroupName1 = "StorageLensGroupName1";
        String storageLensGroupName2 = "StorageLensGroupName2";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevelSelectionCriteria selectionCriteria = new
 StorageLensGroupLevelSelectionCriteria()
                    .withInclude(
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName1,
```

```
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName2);

            System.out.println(selectionCriteria);
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel()
                    .withSelectionCriteria(selectionCriteria);

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

SDK for Java を使用した AWS Organizations の例で Amazon S3 ストレージレンズを使用する

Amazon S3 ストレージレンズを使用して、AWS Organizations の階層に属しているすべてのアカウントから、ストレージのメトリクスと使用状況に関するデータを収集できます。詳細については、AWS Organizations とともに Amazon S3 ストレージレンズを使用するを参照してください。

トピック

- S3 Storage Lens のために組織の信頼されたアクセスを有効にする
- S3 Storage Lens で組織の信頼されたアクセスを無効にする
- S3 Storage Lens のために組織の委任された管理者を登録する
- S3 Storage Lens で委任された Organizations の管理者の登録を解除する

S3 Storage Lens のために組織の信頼されたアクセスを有効にする

Example S3 Storage Lens のために組織の信頼されたアクセスを有効にする

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import com.amazonaws.services.organizations.model.EnableAWSServiceAccessRequest;

public class EnableOrganizationsTrustedAccess {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.enableAWSServiceAccess(new
 EnableAWSServiceAccessRequest()
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
```

```
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

S3 Storage Lens で組織の信頼されたアクセスを無効にする

Example S3 Storage Lens で組織の信頼されたアクセスを無効にする

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import com.amazonaws.services.organizations.model.DisableAWSServiceAccessRequest;

public class DisableOrganizationsTrustedAccess {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            // Make sure to remove any existing delegated administrator for S3 Storage
 Lens
            // before disabling access; otherwise, the request will fail.
            organizationsClient.disableAWSServiceAccess(new
 DisableAWSServiceAccessRequest()
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
```

```
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

S3 Storage Lens のために組織の委任された管理者を登録する

Example S3 Storage Lens のために組織の委任された管理者を登録する

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import
 com.amazonaws.services.organizations.model.RegisterDelegatedAdministratorRequest;

public class RegisterOrganizationsDelegatedAdministrator {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            String delegatedAdminAccountId = "111122223333"; // Account Id for the
 delegated administrator.
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.registerDelegatedAdministrator(new
 RegisterDelegatedAdministratorRequest()
                .withAccountId(delegatedAdminAccountId)
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
```

```
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

S3 Storage Lens で委任された Organizations の管理者の登録を解除する

Example S3 Storage Lens で委任された Organizations の管理者の登録を解除する

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import
 com.amazonaws.services.organizations.model.DeregisterDelegatedAdministratorRequest;

public class DeregisterOrganizationsDelegatedAdministrator {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            String delegatedAdminAccountId = "111122223333"; // Account Id for the
 delegated administrator.
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.deregisterDelegatedAdministrator(new
 DeregisterDelegatedAdministratorRequest()
                .withAccountId(delegatedAdminAccountId)
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
```

```
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
    }
}
```

S3 Storage Lens グループの使用

Amazon S3 ストレージレンズグループは、オブジェクトメタデータに基づくカスタムフィルタを使用してメトリクスを集計します。Storage Lens グループを使用すると、年齢別のオブジェクトの分布や最も一般的なファイルタイプなど、データの特性を掘り下げて調べることができます。たとえば、メトリクスをオブジェクトタグでフィルタリングして急速に増加しているデータセットを特定したり、オブジェクトのサイズと経過時間に基づいてストレージを視覚化してストレージアーカイブ戦略に役立てたりすることができます。結果として、Amazon S3 ストレージレンズグループは、S3 ストレージの理解と最適化に役立ちます。

Storage Lens グループを使用すると、プレフィックス、サフィックス、オブジェクトタグ、オブジェクトサイズ、オブジェクト経過時間などのオブジェクトメタデータを使用して S3 Storage Lens メトリクスを分析およびフィルタリングできます。また、これらのフィルタを組み合わせて適用できます。Storage Lens グループを S3 Storage Lens ダッシュボードにアタッチすると、Amazon S3 ストレージレンズグループによって集計された S3 Storage Lens メトリクスをダッシュボードで直接表示できます。

たとえば、オブジェクトサイズや経過時間層でメトリクスをフィルタリングして、ストレージのどの部分が小さなオブジェクトで構成されているかを判断することもできます。その後、この情報を S3 Intelligent-Tiering や S3 Lifecycle で使用して、小さなオブジェクトをさまざまなストレージクラスに移行することで、コストとストレージの最適化を図ることができます。

トピック

- S3 Storage Lens グループの仕組み
- Storage Lens グループの使用

S3 Storage Lens グループの仕組み

Storage Lens グループを使用すると、オブジェクトメタデータに基づくカスタムフィルタを使用してメトリクスを集約できます。カスタムフィルタを定義する場合、プレフィックス、サフィックス、

オブジェクトタグ、オブジェクトサイズ、オブジェクト経過時間、またはこれらのカスタムフィルタの組み合わせを使用できます。Storage Lens グループの作成時には、1 つのフィルタまたは複数のフィルタ条件を含めることもできます。複数のフィルタ条件を指定するには、`And` または `Or` 論理演算子を使用します。

Storage Lens グループを作成して設定すると、Storage Lens グループ自体が、グループを関連付けるダッシュボードのカスタムフィルタとして機能します。その後、ダッシュボードで Storage Lens グループフィルタを使用して、グループで定義したカスタムフィルタに基づくストレージメトリクスを取得できます。

S3 Storage Lens ダッシュボードで Storage Lens グループのデータを表示するには、グループを作成した後でグループをダッシュボードにアタッチする必要があります。Storage Lens グループを Storage Lens ダッシュボードにアタッチすると、ダッシュボードは 48 時間以内にストレージ使用状況のメトリクスを収集します。その後、このデータを Storage Lens ダッシュボードで視覚化するか、メトリクスのエクスポートを使用して エクスポートできます。Storage Lens グループをダッシュボードにアタッチし忘れると、Storage Lens グループのデータがキャプチャされず、どこにも表示されなくなります。

> **Note**
>
> - S3 Storage Lens グループを作成すると、AWS リソースが作成されます。したがって、各 Storage Lens グループには独自の Amazon リソースネーム (ARN) があり、S3 Storage Lens ダッシュボードにアタッチしたり、S3 Storage Lens ダッシュボードから除外したりするときに指定できます。
> - Storage Lens グループがダッシュボードに関連付けられていない場合、Storage Lens グループの作成に追加料金は発生しません。
> - S3 Storage Lens は、一致するすべての Storage Lens グループのオブジェクトの使用状況メトリクスを集約します。そのため、オブジェクトが 2 つ以上の Storage Lens グループのフィルタ条件に一致すると、ストレージ使用状況全体で同じオブジェクトの数が繰り返し表示されます。

指定したホームリージョン (サポート対象 AWS リージョン のリストから) のアカウントレベルで Storage Lens グループを作成できます。これで、ダッシュボードが同じ AWS アカウント とホームリージョンにある限り、Storage Lens グループを複数の Storage Lens ダッシュボードに接続できます。それぞれの AWS アカウント でホームリージョンごとに最大 50 の Storage Lens グループを作成できます。

AmazonS3 コンソール、AWS Command Line Interface、(AWS CLI)、AWS SDK、または Amazon S3 REST API を使用して S3 Storage Lens グループを作成および管理できます。

トピック

Storage Lens グループ集約メトリクスの表示

グループをダッシュボードにアタッチすることにより、Storage Lens グループの集約されたメトリクスを表示できます。アタッチする Storage Lens グループは、ダッシュボードアカウントの指定されたホームリージョン内に存在している必要があります。

Storage Lens グループをダッシュボードにアタッチするには、ダッシュボード設定の Storage Lens グループ集約セクションでグループを指定する必要があります。複数の Storage Lens グループがある場合、Storage Lens グループ集約結果をフィルタして、目的のグループを含めたり除外したりできます。グループをダッシュボードに関連付ける方法の詳細については、「the section called “Storage Lens グループをアタッチまたは削除する。”」を参照してください。

グループをアタッチした後、48 時間以内にダッシュボードに追加の Storage Lens グループ集約データが表示されます。

> **Note**
>
> Storage Lens グループ集約メトリクスを表示するには、そのグループを S3 Storage Lens ダッシュボードにアタッチする必要があります。

Storage Lens グループのアクセス許可。

Storage Lens グループでは、S3 Storage Lens のアクションへのアクセスを許可するために、AWS Identity and Access Management (IAM) で特定のアクセス許可が必要となります。これらのアクセス許可を付与するには、アイデンティティベースの IAM ポリシーを使用できます。このポリシーを、IAM ユーザー、グループ、またはロールにアタッチして、アクセス許可を付与することができ

ます。このようなアクセス許可には、Storage Lens グループの作成や削除、設定の表示、タグの管理などが含まれます。

アクセス許可を付与する IAM ユーザーまたはロールは、Storage Lens グループを作成または所有するアカウントに属している必要があります。

Storage Lens グループを使用し、Storage Lens グループのメトリクスを表示するには、まず S3 Storage Lens を使用するための適切なアクセス許可が必要です。詳細については、「the section called “S3 ストレージレンズアクセス許可。”」を参照してください。

S3 Storage Lens グループを作成して管理するには、実行するアクションに応じて次の IAM アクセス許可が必要です。

アクション	IAM アクセス許可
新しい Storage Lens グループを作成する	`s3:CreateStorageLensGroup`
タグ付きの新しい Storage Lens グループを作成する	`s3:CreateStorageLensGroup` , `s3:TagResource`
既存の Storage Lens グループを更新する	`s3:UpdateStorageLensGroup`
Storage Lens グループ設定の詳細を返す	`s3:GetStorageLensGroup`
ホームリージョンのすべての Storage Lens グループを一覧表示する	`s3:ListStorageLensGroups`
Storage Lens グループを削除する。	`s3:DeleteStorageLensGroup`
Storage Lens グループに追加されたタグを一覧表示する	`s3:ListTagsForResource`
既存の Storage Lens グループの Storage Lens グループタグを追加または更新する	`s3:TagResource`
Storage Lens グループからタグを削除する	`s3:UntagResource`

Storage Lens グループを作成するアカウントで IAM ポリシーを設定する方法の例を次に示します。このポリシーを使用するには、*`us-east-1`* を Storage Lens グループがあるホームリージョンに置き換えてください。*`111122223333`* を自分の ID と置き換え、AWS アカウント を Storage Lens グ

ループの名前と置き換えます。これらのアクセス許可をすべての Storage Lens グループに適用するには、*example-storage-lens-group* を * と置き換えてください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EXAMPLE-Statement-ID",
            "Effect": "Allow",
            "Action": [
                "s3:CreateStorageLensGroup",
                "s3:UpdateStorageLensGroup",
                "s3:GetStorageLensGroup",
                "s3:ListStorageLensGroups",
                "s3:DeleteStorageLensGroup,
                "s3:TagResource",
                "s3:UntagResource",
                "s3:ListTagsForResource"
                ],
            "Resource": "arn:aws:s3:us-east-1:111122223333:storage-lens-group/example-storage-lens-group"
        }
    ]
}
```

S3 Storage Lens アクセス許可の詳細については、「Amazon S3 ストレージレンズアクセス許可」を参照してください。IAM ポリシー言語については、「Amazon S3 のポリシーとアクセス許可」を参照してください。

Storage Lens グループの設定

S3 Storage Lens グループ名

ダッシュボードにアタッチするグループを簡単に判断できるように、Storage Lens グループには用途を示す名前を付けることをお勧めします。Storage Lens グループをダッシュボードにアタッチするには、ダッシュボード設定の [Storage Lens グループ集約] セクションでグループを指定する必要があります。

Storage Lens グループ名はアカウント内で一意である必要があります。名前は、64 文字以下にし、使用できるのは、文字 (a ~ z、A ~ Z)、数字 (0-9)、ハイフン (-)、およびアンダースコア (_) のみです。

ホームリージョン

ホームリージョンは、Storage Lens グループを作成および管理する AWS リージョン です。Storage Lens グループは、Amazon S3 ストレージレンズダッシュボードと同じホームリージョンに作成されます。Storage Lens グループの設定とメトリクスもこのリージョンに保存されます。ホームリージョンごとに最大 50 の Storage Lens グループを作成できます。

Storage Lens グループを作成した後は、ホームリージョンを編集できません。

スコープ

Storage Lens グループにオブジェクトを含めるには、そのオブジェクトが Amazon S3 ストレージレンズダッシュボードの範囲内にある必要があります。Storage Lens ダッシュボードの範囲は、S3 Storage Lens ダッシュボード設定のダッシュボードスコープに含めたバケットによって決まります。

オブジェクトにさまざまなフィルタを使用して Storage Lens グループの範囲を定義できます。S3 Storage Lens ダッシュボードでこれらの Storage Lens グループのメトリクスを表示するには、オブジェクトが Storage Lens グループに追加したフィルタと一致している必要があります。例えば、Storage Lens グループにプレフィックス `marketing` とサフィックス `.png` が付いたオブジェクトが含まれているが、それらの条件に一致するオブジェクトがないとします。この場合、この Storage Lens グループのメトリクスは毎日のメトリクスのエクスポートでは生成されず、このグループのメトリクスはダッシュボードに表示されません。

フィルタ

S3 Storage Lens グループでは以下のフィルタを使用できます。

- プレフィックス — 含まれるオブジェクトのプレフィックスを指定します。これはオブジェクトキー名の先頭にある文字列です。例えば、[プレフィックス] フィルタの値 `images` には、`images/`、`images-marketing`、`images/production` のいずれかのプレフィックスを持つオブジェクトが含まれます。プレフィックスの最大長は、1,024 バイトです。
- サフィックス — 含まれるオブジェクトのサフィックスを指定します (`.png`、`.jpeg`、`.csv` など)。プレフィックスの最大長は、1,024 バイトです。
- オブジェクトタグ — フィルタの対象となるオブジェクトタグのリストを指定します。タグキーは 128 Unicode 文字、タグ値は 256 Unicode 文字を超過してはいけません。オブジェクトタグ値フィールドを空のままにした場合、S3 Storage Lens グループは、空のタグ値も持つ他のオブジェクトとオブジェクトのみを一致させることに注意してください。

- 経過時間 — 含まれるオブジェクトのオブジェクト経過時間範囲を日数で指定します。整数のみがサポートされます。
- サイズ — 含まれるオブジェクトのオブジェクトサイズをバイト単位で指定します。整数のみがサポートされます。許容される最大値は 5 TB です。

Storage Lens グループのオブジェクトタグ

最大 10 個のオブジェクトタグフィルタを含む Storage Lens グループを作成できます。次の例には、*Marketing-Department* という名前の Storage Lens グループのフィルタとして 2 つのオブジェクトタグのキーと値のペアが含まれています。この例を使用するには、*Marketing-Department* をグループの名前に置き換え、*object-tag-key-1*、*object-tag-value-1* をフィルタリングするオブジェクトタグのキーと値のペアに置き換えます。

```
{
    "Name": "Marketing-Department",
    "Filter": {
     "MatchAnyTag":[
                {
                    "Key": "object-tag-key-1",
                    "Value": "object-tag-value-1"
                },
                {
                    "Key": "object-tag-key-2",
                    "Value": "object-tag-value-2"
                }
            ]
    }
}
```

論理演算子 (**Or** または **And**)

Storage Lens グループに複数のフィルタ条件を含めるには、論理演算子 (And または Or) を使用できます。次の例では、*Marketing-Department* という名前の Storage Lens グループに Prefix、ObjectAge、ObjectSize フィルタを含む And 演算子があります。And 演算子が使用されるため、これらのフィルタ条件をすべて満たすオブジェクトのみが Storage Lens グループのスコープに含まれます。

この例を使用するには、*user input placeholders* をフィルタする値に置き換えます。

```
{
```

```
    "Name": "Marketing-Department",
    "Filter": {
        "And": {
            "MatchAnyPrefix": [
                "prefix-1",
                "prefix-2",
                "prefix-3/sub-prefix-1"
            ],
             "MatchObjectAge": {
                "DaysGreaterThan": 10,
                "DaysLessThan": 60
            },
            "MatchObjectSize": {
                "BytesGreaterThan": 10,
                "BytesLessThan": 60
            }
        }
    }
}
```

> **Note**
>
> フィルタの条件のいずれかに一致するオブジェクトを含める場合は、この例では And 論理演算子を Or 論理演算子に置き換えてください。

AWS リソースタグ

S3 ストレージレンズグループはそれぞれ独自の Amazon AWS リソースネーム (ARN) を持つリソースとみなされます。そのため、Storage Lens グループを設定するときに、AWS オプションでグループにリソースタグを追加できます。Storage Lens グループごとに最大 50 個のタグを追加できます。タグ付きの Storage Lens グループを作成するには、`s3:CreateStorageLensGroup` と `s3:TagResource` の権限が必要です。

AWS リソースタグを使用して、部門、事業内容、またはプロジェクトごとにリソースを分類できます。そうすることで、同じ種類のリソースが多い場合に役立ちます。タグを適用すると、割り当てたタグに基づいて特定の Storage Lens グループをすばやく識別できます。また、コストの追跡と割り当てにもタグを使用できます。

さらに、Storage Lens グループに AWS リソースタグを追加すると、属性ベースのアクセス制御 (ABAC) が有効になります。ABAC は、このケースタグの属性に基づいて権限を定義する認証戦略で

す。IAM ポリシーでリソースタグを指定する条件を使用して、AWS リソースへのアクセスを制御できます。

タグのキーと値は編集でき、タグはリソースからいつでも削除できます。また、次の点について注意してください:

- キーとタグの値は大文字と小文字が区別されます。
- 特定のリソースについて既存のタグと同じキーを持つタグを追加した場合、以前の値は新しい値によって上書きされます。
- リソースを削除すると、リソースのタグも削除されます。
- AWS リソースタグにはプライベートデータや機密データを含めないでください。
- システムタグ (または `aws:` で始まるタグキーを持つタグ) はサポートされていません。
- 各タグキーの長さは 128 文字以下にする必要があります。各タグ値の長さは 256 文字以下にする必要があります。

Storage Lens グループメトリクスのエクスポート

S3 Storage Lens グループメトリクスは、Storage Lens グループがアタッチされているダッシュボードの Amazon S3 ストレージレンズメトリクスエクスポートに含まれています。Storage Lens メトリクスのエクスポート機能に関する一般的な情報については、「データエクスポートで Amazon S3 Storage Lens のメトリクスを確認する」を参照してください。

Storage Lens グループのメトリクスエクスポートには、Storage Lens グループをアタッチしたダッシュボードの範囲内にあるすべての S3 Storage Lens メトリクスが含まれます。エクスポートには、Storage Lens グループの追加メトリクスデータも含まれます。

Storage Lens グループを作成すると、そのグループの所属するダッシュボードのメトリクスエクスポートを設定したときに選択したバケットに、メトリクスのエクスポートが毎日送信されます。最初のエクスポートを受信するに最大 48 時間かかることがあります。

日次エクスポートでメトリクスを生成するには、オブジェクトが Storage Lens グループに含めたフィルタと一致する必要があります。Storage Lens グループに含められたフィルタに一致するオブジェクトがない場合、メトリクスは生成されません。ただし、オブジェクトが 2 つ以上の Storage Lens グループと一致する場合、そのオブジェクトはメトリクスのエクスポートに表示される際、グループごとに別々に表示されます。

Storage Lens グループのメトリクスは、ダッシュボードのメトリクスエクスポートの `record_type` 列で以下のいずれかの値を探すことにより特定できます。

- STORAGE_LENS_GROUP_BUCKET
- STORAGE_LENS_GROUP_ACCOUNT

record_value 列には、Storage Lens グループのリソース ARN が表示されます (例: arn:aws:s3:*us-east-1*:*111122223333*:storage-lens-group/*Marketing-Department*)。

Storage Lens グループの使用

Amazon S3 ストレージレンズグループは、オブジェクトメタデータに基づくカスタムフィルタを使用してメトリクスを集計します。プレフィックス、サフィックス、オブジェクトタグ、オブジェクトサイズ、オブジェクト経過時間使用して S3 Storage Lens メトリクスを分析およびフィルタリングできます。Amazon S3 ストレージレンズのグループを使用して、Amazon S3 バケット間での使用状況を分類することもできます。その結果、S3 ストレージの理解を深め最適化できるようになります。

Storage Lens グループのデータの視覚化を開始するには、まず Storage Lens グループを S3 Storage Lens ダッシュボードにアタッチする必要があります。Storage Lens グループをダッシュボードで管理する必要がある場合は、ダッシュボードの設定を編集できます。アカウントにどの Storage Lens グループがあるかを確認するには、それらのグループを一覧表示できます。どの Storage Lens グループがダッシュボードにアタッチされているかを確認するには、ダッシュボードの [Storage Lens グループ] タブを随時確認できます。既存の Storage Lens グループのスコープを確認または更新するには、その詳細を表示できます。Storage Lens グループを完全に削除することもできます。

権限を管理するために、ユーザー定義の AWS リソースタグを作成して Storage Lens グループに追加できます。AWS リソースタグを使用して、部門、事業内容、またはプロジェクトごとにリソースを分類できます。そうすることで、同じ種類のリソースが多い場合に役立ちます。タグを適用すると、割り当てたタグに基づいて特定の Storage Lens グループをすばやく識別できます。

さらに、Storage Lens グループに AWS リソースタグを追加すると、属性ベースのアクセス制御 (ABAC) が有効になります。ABAC は、このケースタグの属性に基づいて権限を定義する認証戦略です。IAM ポリシーでリソースタグを指定する条件を使用して、AWS リソースへのアクセスを制御できます。

トピック

- Storage Lens グループの作成
- S3 Storage Lens グループをダッシュボードにアタッチまたは削除する、またはダッシュボードから S3 Storage Lens グループを削除する

- Storage Lens グループのデータの視覚化
- Storage Lens グループの更新
- Storage Lens グループによる AWS リソースタグの管理
- すべての Storage Lens グループの一覧表示
- Storage Lens グループ詳細の表示
- Storage Lens グループの削除

Storage Lens グループの作成

以下の例は、Amazon S3 コンソール AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用して Amazon S3 ストレージレンズグループを作成する方法を示しています。

S3 コンソールの使用

Storage Lens グループを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ページ上部にあるナビゲーションバーで、現在表示されている AWS リージョンの名前をクリックします。次に、切り替え先のリージョンを選択します。
3. ナビゲーションペインで、[Storage Lens グループ]を選択します。
4. [Storage Lens グループの作成] を選択します。
5. [全般] で、現在の [ホームリージョン] を表示して、[ストレージレンズグループ名] を入力します。
6. [スコープ] で、Storage Lens グループに適用するためのフィルタを選択します。複数のフィルタを適用するには、フィルタを選択し、AND または OR 論理演算子を選択します。

   - [プレフィックス] フィルタでは、[プレフィックス] を選択し、プレフィックス文字列を入力します。複数のプレフィックスを追加するには、[プレフィックスを追加] を選択します。プレフィックスを削除するには、削除するボリュームの横にある [削除] を選択します。
   - [オブジェクトタグ] フィルタでは、[オブジェクトタグ] を選択し、オブジェクトのキーと値のペアを入力します。次に、[タグを追加] を選択します。タグを削除するには、削除したいタグの横にある[削除] を選択します。
   - [サフィックス] フィルタでは、[サフィックス] を選択し、サフィックス文字列を入力します。複数のサフィックスを追加するには、[サフィックスを追加] を選択します。サフィックスを削除するには、削除したいサフィックスの横にある[削除] を選択します。

- [経過時間] フィルタでは、オブジェクトの有効期間を日単位で指定します。[オブジェクトの最短経過時間を指定] を選択し、オブジェクトの最低経過時間を入力します。次に、[オブジェクトの最長経過時間を指定] を選択し、オブジェクトの最長経過時間を入力します。
- [サイズ] フィルタで、オブジェクトサイズの範囲と測定単位を指定します。[最小オブジェクトサイズを指定] を選択し、オブジェクトの最小オブジェクトサイズを入力します。[最大オブジェクトサイズを指定] を選択し、オブジェクトの最大オブジェクトサイズを入力します。

7. (オプション) AWS リソースタグには、キーと値のペアを追加し、[タグを追加] を選択します。
8. [Storage Lens グループの作成] を選択します。

AWS CLI の使用

次の AWS CLI コマンド例は、Storage Lens グループを作成します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control create-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json
```

次の AWS CLI コマンド例は、2 つの AWS リソースタグを持つ Storage Lens グループを作成します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control create-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json \
--tags Key=k1,Value=v1 Key=k2,Value=v2
```

JSON 設定例については、「Storage Lens グループの設定」を参照してください。

AWS SDK for Java の使用

次の AWS SDK for Java 例は、Storage Lens グループを作成します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Example — 1 つのフィルタで Storage Lens グループを作成します。

次の例は、*Marketing-Department* という Storage Lens グループを作成します。このグループには、経過時間範囲を *30* ~ *90* 日で指定するオブジェクト経過時間フィルタがあります。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;
```

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectAge;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;

public class CreateStorageLensGroupWithObjectAge {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            StorageLensGroupFilter objectAgeFilter = StorageLensGroupFilter.builder()
                    .matchObjectAge(MatchObjectAge.builder()
                            .daysGreaterThan(30)
                            .daysLessThan(90)
                            .build())
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(objectAgeFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
```

```
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example — 複数のフィルタを含む **AND** 演算子を使用して Storage Lens グループを作成します。

次の例は、*Marketing-Department* という Storage Lens グループを作成します。このグループ AND は演算子を使用して、オブジェクトがすべてのフィルタ条件に一致する必要があることを示します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectAge;
import software.amazon.awssdk.services.s3control.model.MatchObjectSize;
import software.amazon.awssdk.services.s3control.model.S3Tag;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupAndOperator;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;


public class CreateStorageLensGroupWithAndFilter {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create object tags.
            S3Tag tag1 = S3Tag.builder()
                    .key("object-tag-key-1")
                    .value("object-tag-value-1")
                    .build();
            S3Tag tag2 = S3Tag.builder()
                    .key("object-tag-key-2")
```

```
                    .value("object-tag-value-2")
                    .build();

            StorageLensGroupAndOperator andOperator =
StorageLensGroupAndOperator.builder()
                    .matchAnyPrefix("prefix-1", "prefix-2", "prefix-3/sub-prefix-1")
                    .matchAnySuffix(".png", ".gif", ".jpg")
                    .matchAnyTag(tag1, tag2)
                    .matchObjectAge(MatchObjectAge.builder()
                            .daysGreaterThan(30)
                            .daysLessThan(90).build())
                    .matchObjectSize(MatchObjectSize.builder()
                            .bytesGreaterThan(1000L)
                            .bytesLessThan(6000L).build())
                    .build();

            StorageLensGroupFilter andFilter = StorageLensGroupFilter.builder()
                    .and(andOperator)
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(andFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
```

```
    }
}
```

Example — 複数のフィルタを含む **OR** 演算子を使用して Storage Lens グループを作成します。

次の例は、*Marketing-Department* という Storage Lens グループを作成します。このグループは、OR 演算子を使用してプレフィックスフィルタ (*prefix-1*、*prefix-2*、*prefix3/sub-prefix-1*) または *1000* ~ *6000* バイトまでのサイズ範囲を持つオブジェクトサイズフィルタを適用します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectSize;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupOrOperator;

public class CreateStorageLensGroupWithOrFilter {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            StorageLensGroupOrOperator orOperator =
 StorageLensGroupOrOperator.builder()
                    .matchAnyPrefix("prefix-1", "prefix-2", "prefix-3/sub-prefix-1")
                    .matchObjectSize(MatchObjectSize.builder()
                            .bytesGreaterThan(1000L)
                            .bytesLessThan(6000L)
                            .build())
                    .build();

            StorageLensGroupFilter orFilter = StorageLensGroupFilter.builder()
                    .or(orOperator)
                    .build();
```

```
            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(orFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example — 1 つのフィルタと 2 つの AWS リソースタグを持つ Storage Lens グループを作成します。

次の例は、サフィックスフィルタを持つ *Marketing-Department* という Storage Lens グループを作成します。この例では、Storage Lens グループに 2 AWS つのリソースタグも追加しています。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
```

```
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.Tag;

public class CreateStorageLensGroupWithResourceTags {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create AWS resource tags.
            Tag resourceTag1 = Tag.builder()
                    .key("resource-tag-key-1")
                    .value("resource-tag-value-1")
                    .build();
            Tag resourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-value-2")
                    .build();

            StorageLensGroupFilter suffixFilter = StorageLensGroupFilter.builder()
                    .matchAnySuffix(".png", ".gif", ".jpg")
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(suffixFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .tags(resourceTag1, resourceTag2)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
```

```
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

JSON 設定例については、「Storage Lens グループの設定」を参照してください。

S3 Storage Lens グループをダッシュボードにアタッチまたは削除する、またはダッシュボードから S3 Storage Lens グループを削除する

Amazon S3 Storage Lens の高度なティアにアップグレードしたら、Storage Lens グループをダッシュボードにアタッチできます。複数の Storage Lens グループがある場合、目的のグループを含めたり除外したりできます。

Storage Lens グループは、ダッシュボードアカウントの指定されたホームリージョン内に存在している必要があります。Storage Lens グループをダッシュボードに接続すると、48 時間以内にメトリクスのエクスポートで追加の Storage Lens グループの集計データを受け取ります。

> **Note**
>
> Storage Lens グループの集計メトリクスを表示するには、そのデータを Storage Lens ダッシュボードにアタッチする必要があります。Storage Lens グループの JSON 設定ファイルの例については、「JSON の Storage Lens グループを使用した S3 Storage Lens の設定例」を参照してください。

S3 Storage Lens ダッシュボードに Storage Lens のグループをアタッチする

Storage Lens グループを Storage Lens ダッシュボードにアタッチするには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. 左ナビゲーションペインの、[S3 Storage Lens] で、[ダッシュボード] を選択します。
3. Storage Lens グループをアタッチする Storage Lens ダッシュボードのオプションボタンを選択します。
4. [Edit] (編集) を選択します。

5. メトリクスの選択で [Advanced metrics and recommendations] (アドバンストメトリクスとレコメンデーション) を選択します。
6. [Storage Lens グループ集計] を選択します。

> **Note**
>
> デフォルトでは、[高度なメトリクス] も選択されています。ただし、Storage Lens グループのデータを集計する必要がないため、この設定の選択を解除することもできます。

7. [Storage Lens グループ集計] までスクロールし、データ集計に含める、または除外する 1 つまたは複数の Storage Lens グループを指定します。次のフィルタリングオプションを設定できます。

   - 特定の Storage Lens グループを含める場合は、[Storage Lens グループを含める] を選択します。[含めるStorage Lens グループ] で、Storage Lens グループを選択します。
   - すべての Storage Lens グループを含める場合は、[ホームリージョンのすべてのStorage Lens グループをこのアカウントに含める] を選択します。
   - 特定の Storage Lens グループを除外する場合は、[Storage Lens グループを除外する] を選択します。[除外する Storage Lens グループ] で、除外する Storage Lens グループを選択します。
8. [Save changes] (変更の保存) をクリックします。Storage Lens グループを正しく設定していれば、48 時間以内にダッシュボードに追加の Storage Lens グループ集計データが表示されます。

S3 Storage Lens ダッシュボードからの Storage Lens グループの削除

S3 Storage Lens ダッシュボードから Storage Lens グループを削除するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインの、[S3 Storage Lens] で、[ダッシュボード] を選択します。
3. Storage Lens グループを削除する Storage Lens ダッシュボードのオプションボタンを選択します。
4. [ダッシュボード設定を表示] を選択します。
5. [Edit] (編集) を選択します。
6. [メトリクスの選択] セクションまで下にスクロールします。

7. [Storage Lens グループ集計] で、削除する Storage Lens グループの横にある X を選択します。これにより、Storage Lens グループが削除されます。

   すべての Storage Lens グループをダッシュボードに含めた場合は、[このアカウントにホームリージョンのすべての Storage Lens グループを含める] の横にあるチェックボックスをオフにします。
8. [Save changes] (変更の保存) をクリックします。

> **Note**
>
> ダッシュボードに設定の更新が反映されるまでに最大 48 時間かかることがあります。

Storage Lens グループのデータの視覚化

Storage Lens グループのデータは、Amazon S3 ストレージレンズダッシュボードにグループをアタッチすることで視覚化できます。ダッシュボード設定の Storage Lens グループ集計に Storage Lens グループを含めてから、Storage Lens グループのデータがダッシュボードに表示されるまでに最大 48 時間かかることがあります。

ダッシュボードの設定が更新されると、新しくアタッチされた Storage Lens グループが [Storage Lens グループ] タブの使用可能なリソースのリストに表示されます。データを別の次元でスライスすることで、[概要] タブでストレージ使用状況をさらに分析することもできます。たとえば、上位 3 つのカテゴリにリストされている項目の 1 つを選択し、[分析基準] を選択すると、データを別のディメンションで分割できます。フィルタ自体と同じディメンションを適用することはできません。

> **Note**
>
> Storage Lens グループフィルタとプレフィックスフィルタ、またはその逆は適用できません。また、プレフィックスフィルタを使用して Storage Lens グループをさらに分析することもできません。

Amazon S3 ストレージレンズダッシュボードの [Storage Lens グループ] タブを使用して、ダッシュボードにアタッチされている Storage Lens グループのデータ視覚化をカスタマイズできます。ダッシュボードに添付されている一部の Storage Lens グループ、またはすべてのデータを視覚化できます。

S3 Storage Lens のデータを視覚化するときは、次の点に注意してください。

- S3 Storage Lens は、一致するすべての Storage Lens グループのオブジェクトの使用状況メトリクスを集計します。そのため、オブジェクトが 2 つ以上の Storage Lens グループのフィルタ条件に一致すると、ストレージ使用状況全体で同じオブジェクトの数が繰り返し表示されます。
- オブジェクトは、Storage Lens グループに含めるフィルタと一致する必要があります。Storage Lens グループに含めるフィルタに一致するオブジェクトがない場合、メトリクスは生成されません。未割り当てのオブジェクトがないかどうかを確認するには、ダッシュボードでアカウントレベルとバケットレベルの合計オブジェクト数を確認します。

Storage Lens グループの更新

以下の例は、Amazon S3 ストレージレンズグループを更新する方法を示しています。Amazon S3 コンソール AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用して Storage Lens グループを更新できます。

S3 コンソールの使用

Storage Lens グループを更新するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens グループ]を選択します。
3. [Storage Lens グループ] で、更新する Storage Lens グループを選択します。
4. [スコープ] で、[編集] を選択します。
5. [スコープ] ページで、Storage Lens グループに適用するためのフィルタを選択します。複数のフィルタを適用するには、フィルタを選択し、AND または OR の論理演算子を選択します。
   - [プレフィックス] フィルタでは、[プレフィックス] を選択し、プレフィックス文字列を入力します。複数のプレフィックスを追加するには、[プレフィックスを追加] を選択します。プレフィックスを削除するには、削除するボリュームの横にある [削除] を選択します。
   - [オブジェクトタグ] フィルタでは、オブジェクトのキーと値のペアを入力します。次に、[タグの追加] を選択します。タグを削除するには、削除したいタグの横にある[削除] を選択します。
   - [サフィックス] フィルタでは、[サフィックス] を選択し、サフィックス文字列を入力します。複数のサフィックスを追加するには、[サフィックスを追加] を選択します。サフィックスを削除するには、削除したいサフィックスの横にある[削除] を選択します。

- [経過時間] フィルタでは、オブジェクトの有効期間を日単位で指定します。[オブジェクトの最短経過時間を指定] を選択し、オブジェクトの最低経過時間を入力します。[オブジェクトの最長経過時間を指定] を選択し、オブジェクトの最長経過時間を入力します。
- [サイズ] フィルタで、オブジェクトサイズの範囲と測定単位を指定します。[最小オブジェクトサイズを指定] を選択し、オブジェクトの最小オブジェクトサイズを入力します。[最大オブジェクトサイズを指定] に、オブジェクトの最大オブジェクトサイズを入力します。

6. [Save changes] (変更の保存) をクリックします。Storage Lens グループの詳細ページが表示されます。
7. (オプション) 新しい AWS リソースタグを追加する場合は、[AWS リソースタグ] セクションまでスクロールし、[タグを追加] を選択します。タグの追加 ページが表示されます。

   新しいキーと値のペアを追加し、[変更の保存] を選択します。Storage Lens グループの詳細ページが表示されます。
8. (オプション) 既存の AWS リソースタグを削除する場合は、[AWS リソースタグ] セクションまでスクロールし、リソースタグを選択します。その後、[Delete] (削除) をクリックします。[AWS タグの削除] ダイアログボックスが表示されます。

   AWS リソースタグを完全に削除するには、もう一度 [削除] を選択します。

> **Note**
>
> AWS リソースタグを完全に削除すると、元に戻すことはできません。

AWS CLI を使用する場合

以下の AWS CLI 例コマンドでは、*marketing-department* という名前の Storage Lens グループの設定の詳細を返します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control get-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

次の AWS CLI 例は、Storage Lens グループを更新します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control update-storage-lens-group --account-id 111122223333 \
```

```
--region us-east-1 --storage-lens-group=file://./marketing-department.json
```

JSON 設定例については、「Storage Lens グループの設定」を参照してください。

AWS SDK for Java の使用

以下の AWS SDK for Java 例では、アカウント *111122223333* の *Marketing-Department* Storage Lens グループの設定の詳細を返します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupResponse;

public class GetStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            GetStorageLensGroupRequest getRequest =
 GetStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            GetStorageLensGroupResponse response =
 s3ControlClient.getStorageLensGroup(getRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
```

```
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

以下の例では、アカウント *111122223333* で *Marketing-Department* という名前の Storage Lens グループを更新します。この例では、*.png*、*.gif*、*.jpg*、または *.jpeg* のいずれかのサフィックスに一致するオブジェクトを含むようにダッシュボードスコープを更新します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.UpdateStorageLensGroupRequest;

public class UpdateStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create updated filter.
            StorageLensGroupFilter suffixFilter = StorageLensGroupFilter.builder()
                    .matchAnySuffix(".png", ".gif", ".jpg", ".jpeg")
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(suffixFilter)
                    .build();

            UpdateStorageLensGroupRequest updateStorageLensGroupRequest =
 UpdateStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId)
```

```
                    .build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.updateStorageLensGroup(updateStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

JSON 設定例については、「Storage Lens グループの設定」を参照してください。

Storage Lens グループによる AWS リソースタグの管理

Amazon S3 ストレージレンズグループはそれぞれ独自の Amazon AWS リソースネーム (ARN) を持つリソースとみなされます。そのため、Storage Lens グループを設定するときに、AWS オプションでグループにリソースタグを追加できます。Storage Lens グループ当たり最大 50 個のタグを追加できます。タグ付きの Storage Lens グループを作成するには、`s3:CreateStorageLensGroup` と `s3:TagResource` の権限が必要です。

AWS リソースタグを使用して、部門、事業内容、またはプロジェクトごとにリソースを分類できます。そうすることで、同じ種類のリソースが多い場合に役立ちます。タグを適用すると、割り当てたタグに基づいて特定の Storage Lens グループをすばやく識別できます。また、コストの追跡と割り当てにもタグを使用できます。

さらに、Storage Lens グループに AWS リソースタグを追加すると、属性ベースのアクセス制御 (ABAC) が有効になります。ABAC は、このケースタグの属性に基づいて権限を定義する認証戦略です。IAM ポリシーでリソースタグを指定する条件を使用して、AWS リソースへのアクセスを制御できます。

タグのキーと値は編集でき、タグはリソースからいつでも削除できます。また、次の点について注意してください:

- キーとタグの値は大文字と小文字が区別されます。
- 特定のリソースについて既存のタグと同じキーを持つタグを追加した場合、以前の値は新しい値によって上書きされます。
- リソースを削除すると、リソースのタグも削除されます。
- AWS リソースタグにはプライベートデータや機密データを含めないでください。
- システムタグ (aws: で始まるタグキーの場合) はサポートされていません。
- 各タグキーの長さは 128 文字以下にする必要があります。各タグ値の長さは 256 文字以下にする必要があります。

次の例は、Storage Lens グループで AWS リソースタグを使用する方法を示しています。

トピック

- Storage Lens グループに AWS リソースタグを追加する
- Storage Lens グループのタグ値の更新
- Storage Lens グループから AWS リソースタグを削除する
- Storage Lens グループのタグの一覧表示

Storage Lens グループに AWS リソースタグを追加する

次の例は、Amazon Storage Lens グループに AWS リソースタグを使用する方法を示しています。Amazon S3 コンソール AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用してリソースタグを追加できます。

S3 コンソールの使用

Storage Lens グループに AWS リソースタグを追加するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens グループ] を選択します。
3. [Storage Lens グループ] で、更新する Storage Lens グループを選択します。
4. [AWS リソースタグ] で [タグを追加] を選択します。
5. [タグの追加] ページで、新しいキーと値のペアを追加します。

> **Note**
>
> 既存のタグと同じ新しいタグを追加すると、以前のタグ値は上書きされます。

6. (オプション) 複数の新しいタグを追加するには、[タグを追加] を再度選択して新しいエントリを追加し続けます。Storage Lens グループには、最大 50 個の AWS リソースタグを追加できます。
7. (オプション) 新規追加されたエントリを削除する場合は、削除するタグの横にある[削除] を選びます。
8. [Save changes] (変更の保存) をクリックします。

AWS CLI を使用する場合

以下の AWS CLI コマンド例では、*marketing-department* という名前の既存の Storage Lens グループに 2 つのリソースタグを追加しています。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control tag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1 --tags Key=k1,Value=v1 Key=k2,Value=v2
```

AWS SDK for Java の使用

以下の AWS SDK for Java 例では、既存の Storage Lens グループに 2 つの AWS リソースタグを追加しています。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;
```

```
public class TagResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            Tag resourceTag1 = Tag.builder()
                .key("resource-tag-key-1")
                .value("resource-tag-value-1")
                .build();
            Tag resourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-value-2")
                    .build();
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tags(resourceTag1, resourceTag2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.tagResource(tagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Storage Lens グループのタグ値の更新

以下の例は、Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用して Storage Lens グループを更新する方法を示しています。

S3 コンソールの使用

Storage Lens グループの AWS リソースタグを更新するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens グループ] を選択します。
3. [Storage Lens グループ] で、更新する Storage Lens グループを選択します。
4. [AWSリソースタグ] で、更新するタグを選択します。
5. 更新するキーと値のペアの同じキーを使用して、新しいタグ値を追加します。チェックマークアイコンを選択してタグの値を更新します。

   > **Note**
   >
   > 既存のタグと同じ新しいタグを追加すると、以前のタグ値は上書きされます。

6. (オプション) 新しいタグを追加する場合は、[タグの追加] を選択して新しいエントリを追加します。タグの追加 ページが表示されます。

   Storage Lens グループには、最大 50 個の AWS リソースタグを追加できます。新しいタグの追加を完了したら、[変更の保存] を選択します。
7. (オプション) 新規追加されたエントリを削除する場合は、削除するタグの横にある[削除] を選びます。タグの削除を完了したら、[タグの保存] を選択します。

AWS CLI を使用する場合

次の AWS CLI コマンド例は、*marketing-department* という名前の Storage Lens グループの 2 つのタグ値を更新します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control tag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1 --tags Key=k1,Value=v3 Key=k2,Value=v4
```

AWS SDK for Java の使用

次の AWS SDK for Java 例では、2 つの Storage Lens グループのタグ値を更新します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;

public class UpdateTagsForResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            Tag updatedResourceTag1 = Tag.builder()
                .key("resource-tag-key-1")
                .value("resource-tag-updated-value-1")
                .build();
            Tag updatedResourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-updated-value-2")
                    .build();
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tags(updatedResourceTag1, updatedResourceTag2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.tagResource(tagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
```

```
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Storage Lens グループから AWS リソースタグを削除する

次の例は、Storage Lens グループから AWS リソースタグを削除する方法を示しています。Amazon S3 コンソール AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用してタグを削除できます。

S3 コンソールの使用

Storage Lens グループから AWS リソースタグを削除するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens グループ] を選択します。
3. [Storage Lens グループ] で、更新する Storage Lens グループを選択します。
4. [AWSリソースタグ] で、削除するキーと値のペアを選択します。
5. [Delete] (削除) をクリックします。[AWSリソースタグの削除] ダイアログボックスが表示されます。

> **Note**
>
> タグを使用してアクセスを制御すると、関連するリソースに影響を与える可能性があります。タグを完全に削除すると、元に戻すことはできません。

6. [削除] を選択してキー値のペアを完全に削除します。

AWS CLI を使用する場合

次の AWS CLI コマンドは、既存の Storage Lens グループから 2 つの AWS リソースタグを削除します。このコマンド例を使用するには、*user input placeholders* を独自の情報に置き換えてください。

```
aws s3control untag-resource --account-id 111122223333 \
```

```
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/Marketing-
Department \
--region us-east-1 --tag-keys k1 k2
```

AWS SDK for Java の使用

次の AWS SDK for Java 例では、アカウント *111122223333* で指定した Storage Lens グループの Amazon リソースネーム (ARN) から 2 つの AWS リソースタグを削除します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.UntagResourceRequest;

public class UntagResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            String tagKey1 = "resource-tag-key-1";
            String tagKey2 = "resource-tag-key-2";
            UntagResourceRequest untagResourceRequest = UntagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tagKeys(tagKey1, tagKey2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.untagResource(untagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
```

```
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Storage Lens グループのタグの一覧表示

以下の例は、Storage Lens グループに関連付けられている AWS リソースタグを一覧表示する方法を示しています。Amazon S3 コンソール AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用してタグを一覧表示できます。

S3 コンソールの使用

Storage Lens グループのタグとタグ値のリストを確認するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens グループ]を選択します。
3. [Storage Lens グループ] で、対象となる Storage Lens グループを選択します。
4. [AWS リソースタグ] セクションまでスクロールダウンします。Storage Lens グループに追加されたすべてのユーザー定義 AWS リソースタグが、そのタグ値とともに一覧表示されます。

AWS CLI を使用する場合

次の AWS CLI コマンド例では、*marketing-department* という名前の Storage Lens グループの Storage Lens グループタグすべてを一覧表示しています。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control list-tags-for-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1
```

AWS SDK for Java の使用

次の AWS SDK for Java 例では、指定した Storage Lens グループ Amazon リソースネーム (ARN) のStorage Lens グループタグ値を一覧表示しています。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceRequest;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceResponse;

public class ListTagsForResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            ListTagsForResourceResponse response =
 s3ControlClient.listTagsForResource(listTagsForResourceRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

すべての Storage Lens グループの一覧表示

以下の例は、AWS アカウント およびホームリージョン内のすべての Amazon S3 ストレージレンズグループを一覧表示する方法を示しています。これらの例は、Amazon S3 コンソール AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用してすべての Storage Lens グループを一覧表示する方法を示しています。

S3 コンソールの使用

アカウントとホームリージョンのすべての Amazon S3 ストレージレンズグループを一覧表示するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens グループ] を選択します。
3. Storage Lens グループには、アカウント内の Storage Lens グループのリストが表示されます。

AWS CLI を使用する場合

次の AWS CLI 例では、アカウントのすべての Storage Lens グループを一覧表示しています。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control list-storage-lens-groups --account-id 111122223333 \
--region us-east-1
```

AWS SDK for Java の使用

次の AWS SDK for Java 例では、アカウント *111122223333* の Storage Lens グループを一覧表示しています。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListStorageLensGroupsRequest;
```

```
import software.amazon.awssdk.services.s3control.model.ListStorageLensGroupsResponse;

public class ListStorageLensGroups {
    public static void main(String[] args) {
        String accountId = "111122223333";

        try {
            ListStorageLensGroupsRequest listStorageLensGroupsRequest =
 ListStorageLensGroupsRequest.builder()
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            ListStorageLensGroupsResponse response =
 s3ControlClient.listStorageLensGroups(listStorageLensGroupsRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Storage Lens グループ詳細の表示

以下の例は、Amazon S3 ストレージレンズグループ設定の詳細を表示する方法を示しています。Amazon S3 コンソール、AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用してこれらの詳細を表示できます。

S3 コンソールの使用

Storage Lens グループ設定の詳細を表示するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。

2. ナビゲーションペインで、[Storage Lens グループ]を選択します。
3. [Storage Lens グループ] で、対象となる Storage Lens グループの横にあるオプションボタンを選択します。
4. [View details] (詳細を表示する) を選択します。これで、Storage Lens グループの詳細を確認できます。

AWS CLI を使用する場合

次の AWS CLI 例では、Storage Lens グループの設定の詳細を返します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control get-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

AWS SDK for Java の使用

以下の AWS SDK for Java 例では、アカウント *111122223333* の *Marketing-Department* という Storage Lens グループの設定の詳細を返します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupResponse;

public class GetStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            GetStorageLensGroupRequest getRequest =
 GetStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
```

```
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            GetStorageLensGroupResponse response =
 s3ControlClient.getStorageLensGroup(getRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Storage Lens グループの削除

以下の例は、Amazon S3 コンソール AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用して Amazon S3 ストレージレンズグループを削除する方法を示しています。

S3 コンソールの使用

Storage Lens グループを削除するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. ナビゲーションペインで、[Storage Lens グループ] を選択します。
3. [Storage Lens グループ] で、削除する Storage Lens グループの横にあるオプションボタンを選択します。
4. [Delete] (削除) をクリックします。[Storage Lens グループの削除] ダイアログボックスが表示されます。
5. もう一度 [削除] を選択すると、Storage Lens グループが完全に削除されます。

> **Note**
>
> Storage Lens グループを削除すると、復元することはできません。

AWS CLI を使用する場合

以下の AWS CLI 例は、*marketing-department* という名前の Storage Lens グループを削除します。このコマンドの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control delete-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

AWS SDK for Java の使用

以下の AWS SDK for Java 例は、アカウント *111122223333* で *Marketing-Department* という名前の Storage Lens グループを削除します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.DeleteStorageLensGroupRequest;

public class DeleteStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            DeleteStorageLensGroupRequest deleteStorageLensGroupRequest =
 DeleteStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.deleteStorageLensGroup(deleteStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
```

```
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

を使用した Amazon S3 リクエストのトレースAWS X-Ray

AWS X-Ray は、アプリケーションが提供するリクエストのデータを収集します。そこからデータを確認し、フィルタリングを行うことで、分散アプリケーションとマイクロサービスアーキテクチャのパフォーマンスの問題とエラーを特定し、トラブルシューティングを行えます。アプリケーションに対するトレース対象のリクエストの場合、リクエスト、レスポンス、およびアプリケーションが ダウンストリーム AWS リソース、マイクロサービス、データベース、HTTP ウェブ API に対して行う呼び出しの、詳細な情報を表示できます。

詳細については、『AWS X-Ray デベロッパーガイド』の「AWS X-Ray とは」を参照してください。

トピック

- X-Ray が Amazon S3 とどのように連携するか
- 利用できるリージョン

X-Ray が Amazon S3 とどのように連携するか

AWS X-Ray は Amazon S3 のトレースコンテキスト伝達をサポートしているため、アプリケーション全体に伝わっていく、エンドツーエンドのリクエストを表示できます。X-Ray は、Amazon S3、AWS Lambda、Amazon EC2 などのサービスがそれぞれ生成するデータと、アプリケーションを構成する多くのリソースを集約します。これにより、どのようにアプリケーションが実行されているのか、全体的に見渡すことができます。

Amazon S3 は X-Ray と統合して、トレースコンテキストを伝達し、アップストリームノードとダウンストリームノードを持つ 1 つのリクエストチェーンを提供します。アップストリームサービスに S3 リクエストと有効な形式のトレースヘッダーが含まれている場合、Lambda、Amazon SQS、Amazon SNS などのダウンストリームサービスに Amazon S3 がイベント通知を配信するときに、トレースヘッダーを渡します。これらすべてのサービスを X-Ray にアクティブに統合する

と、サービスが 1 つのリクエストチェーンに結び付けられ、Amazon S3 リクエスト全体の詳細を確認できます。

Amazon S3 経由で X-Ray トレースヘッダーを送信するには、リクエストにフォーマット済みの X-Amzn-Trace-ID を含める必要があります。AWS X-Ray SDK を使用して Amazon S3 クライアントを計測することもできます。サポートされている SDK のリストについては、「AWS X-Ray のドキュメント」を参照してください。

サービスマップ

X-Ray サービスマップは、Amazon S3 と他の AWS サービス、およびアプリケーション内のリソースとの関係を、ほぼリアルタイムに表示します。X-Ray サービスマップでエンドツーエンドのリクエストを表示するには、X-Ray コンソールを使用して、Amazon S3 と、アプリケーションが使用している他のサービスとの間の接続マップを表示します。高いレイテンシーが発生している場所を簡単に検出し、これらのサービスのノード分布を視覚化してから、アプリケーションのパフォーマンスに影響を与えている特定のサービスとパスをドリルダウンできます。

X-Ray による分析

X-Ray Analytics コンソールを使用して、トレースを分析し、レイテンシーや失敗率などのメトリクスを表示し、問題の特定とトラブルシューティングに役立つ Insights を生成することもできます。このコンソールには、平均レイテンシーや失敗率などのメトリックスも表示されます。詳細については、「AWS X-Ray デベロッパーガイド」の「AWS X-Ray コンソール」を参照してください。

利用できるリージョン

Amazon S3 の AWS X-Ray サポートは、すべての AWS X-Ray リージョンでご利用いただけます。詳細については、「AWS X-Ray デベロッパーガイド」の「Amazon S3 と AWS X-Ray」を参照してください。

Amazon S3 を使用して静的ウェブサイトをホスティングする

Amazon S3 を使用して、静的ウェブサイトをホストできます。静的ウェブサイトでは、個々のウェブページの内容は静的コンテンツです。ほかに、クライアント側スクリプトが含まれていることもあります。

対照的に、動的ウェブサイトはサーバー側処理に依存しており、例えば、サーバー側スクリプト (PHP、JSP、ASP.NET など) が使用されます。Amazon S3 はサーバーサイドスクリプトをサポートしていませんが、AWS には動的ウェブサイトをホストするための他のリソースがあります。AWS でのウェブサイトホスティングの詳細については、「ウェブホスティング」を参照してください。

> **Note**
>
> AWS Amplify コンソールを使用して、単一ページのウェブアプリケーションをホストできます。AWS Amplify コンソールは、単一ページのアプリケーションフレームワーク (React JS、Vue JS、Angular JS、Nuxt など) と静的サイトジェネレータ (Gatsby JS、React-static、Jekyll、Hugo など) で構築された単一ページアプリケーションをサポートします。詳細については、AWS Amplify コンソールユーザーガイドの開始方法を参照してください。Amazon S3 ウェブサイトエンドポイントは HTTPS をサポートしていません。HTTPS を使用する場合は、Amazon CloudFront を使用して Amazon S3 でホストされている静的ウェブサイトを提供できます。詳細については、CloudFront を使用して Amazon S3 バケットに対する HTTPS リクエストを処理するにはどうすればよいですか? を参照してください。カスタムドメインで HTTPS を使用するには、Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定を参照してください。

Amazon S3 での静的ウェブサイトのホストに関する詳細 (手順やステップバイステップのチュートリアルなど) については、以下のトピックを参照してください。

トピック

- ウェブサイトエンドポイント
- ウェブサイトのホスティングの有効化
- インデックスドキュメントの設定
- カスタムエラードキュメントの設定
- ウェブサイトアクセスのアクセス許可の設定

- (オプション) ウェブトラフィックのログ記録
- (オプション) ウェブページリダイレクトの設定

ウェブサイトエンドポイント

バケットを静的ウェブサイトとして設定すると、そのウェブサイトは、バケットの AWS リージョン固有のウェブサイトエンドポイントで利用できます。ウェブサイトエンドポイントは、REST API リクエストを送信するエンドポイントとは別のものです。エンドポイントの違いの詳細については、「ウェブサイトエンドポイントと REST API エンドポイントの主な違い」を参照してください。

使用しているリージョンに応じて、Amazon S3 ウェブサイトエンドポイントは以下の 2 つの形式のいずれかになります。

- s3-website ダッシュ (-) リージョン - `http://bucket-name.s3-website-Region.amazonaws.com`
- s3-website ドット (.) リージョン -`http://bucket-name.s3-website.Region.amazonaws.com`

これらの URL にアクセスすると、ウェブサイト用に設定したデフォルトのインデックスドキュメントが返されます。Amazon S3 ウェブサイトエンドポイントの完全なリストについては、「Amazon S3 ウェブサイトエンドポイント」を参照してください。

> **Note**
>
> Amazon S3 の静的ウェブサイトのセキュリティを強化するために、Amazon S3 ウェブサイトのエンドポイントドメイン (s3-website-us-east-1.amazonaws.com や s3-website.ap-south-1.amazonaws.com など) はパブリックサフィックスリスト (PSL) に登録されています。セキュリティ強化のため、Amazon S3 統計 ウェブサイトのドメイン名に機密な Cookie を設定する必要が生じた場合は、__Host- プレフィックスの付いた Cookie の使用をお勧めします。このプラクティスは、クロスサイトリクエストフォージェリ (CSRF) 攻撃からドメインを防ぐ際に役立ちます。詳細については、Mozilla 開発者ネットワークの「Set-Cookie」ページを参照してください。

ウェブサイトを公開したい場合、顧客がウェブサイトエンドポイントにあるコンテンツにアクセスできるようにするには、すべてのコンテンツがパブリックに読み取り可能になっている必要があります。詳細については、「ウェブサイトアクセスのアクセス許可の設定」を参照してください。

> **Important**
>
> Amazon S3 ウェブサイトエンドポイントは HTTPS またはアクセスポイントをサポートしていません。HTTPS を使用する場合は、Amazon CloudFront を使用して Amazon S3 でホストされている静的ウェブサイトを提供できます。詳細については、CloudFront を使用して Amazon S3 バケットに対する HTTPS リクエストを処理するにはどうすればよいですか? を参照してください。カスタムドメインで HTTPS を使用するには、Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定を参照してください。
>
> 「リクエスタ支払い」バケットに、ウェブサイトエンドポイントを介してアクセスすることはできません。このようなバケットへのリクエストに対しては 403 Access Denied レスポンスを受け取ります。詳細については、「ストレージ転送と使用量のリクエスタ支払いバケットの使用」を参照してください。

トピック

ウェブサイトエンドポイントの例

以下の例では、静的ウェブサイトとして設定されている Amazon S3 バケットにアクセスする方法を示しています。

Example — ルートレベルでのオブジェクトのリクエスト

バケットのルートレベルに保存されている特定のオブジェクトをリクエストするには、以下の URL 構造を使用します。

```
http://bucket-name.s3-website.Region.amazonaws.com/object-name
```

たとえば、以下の URL は、バケットのルートレベルに保存されている photo.jpg オブジェクトをリクエストするものです。

```
http://example-bucket.s3-website.us-west-2.amazonaws.com/photo.jpg
```

Example — プレフィックスでのオブジェクトのリクエスト

バケット内のフォルダに保存されているオブジェクトをリクエストするには、以下の URL 構造を使用します。

```
http://bucket-name.s3-website.Region.amazonaws.com/folder-name/object-name
```

以下の URL は、バケット内の docs/doc1.html オブジェクトをリクエストするものです。

```
http://example-bucket.s3-website.us-west-2.amazonaws.com/docs/doc1.html
```

DNS CNAME の追加

登録済みのドメインをお持ちの場合、Amazon S3 ウェブサイトエンドポイントを指し示す DNS CNAME エントリを追加することができます。たとえば、ドメイン www.example-bucket.com を登録した場合は、バケット www.example-bucket.com を作成し、www.example-bucket.com.s3-website.*Region*.amazonaws.com を参照する DNS CNAME エントリを追加できます。http://www.example-bucket.com へのリクエストはすべて、www.example-bucket.com.s3-website.*Region*.amazonaws.com にルーティングされます。

詳細については、「CNAME レコードを使用した Amazon S3 URL のカスタマイズ」を参照してください。

Route 53 でのカスタムドメインの使用

Amazon S3 ウェブサイトエンドポイントを使用してウェブサイトにアクセスする代わりに、Amazon Route 53 に登録された独自のドメイン (example.com など) を使用して、コンテンツを提供できます。ルートドメインでウェブサイトをホストするために、Amazon S3 を Route 53 とともに使用することができます。例えば、ルートドメインが example.com で、ウェブサイトを Amazon S3 でホスティングする場合に、そのウェブサイトにブラウザからアクセスするには「http://www.example.com」または「http://example.com」を入力します。

チュートリアル例については、「チュートリアル: Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定」を参照してください。

ウェブサイトエンドポイントと REST API エンドポイントの主な違い

Amazon S3 ウェブサイトエンドポイントは、ウェブブラウザを介してアクセスするように最適化されています。以下の表は、REST API エンドポイントとウェブサイトエンドポイントの主な違いをまとめたものです。

主な違い	REST API エンドポイント	ウェブサイトエンドポイント
アクセスコントロール	パブリックコンテンツとプライベートコンテンツの両方をサポートします。	公開で読み取り可能なコンテンツのみをサポートします。
エラーメッセージの処理	XML 形式のエラーレスポンスを返します。	HTML ドキュメントを返します。
リダイレクトのサポート	該当しません	オブジェクトレベルとバケットレベルの両方のリダイレクトをサポートします。
サポートされるリクエスト	バケットおよびオブジェクトのすべてのオペレーションをサポートします。	オブジェクトに対しては GET リクエストと HEAD リクエストのみをサポートします。
バケットのルートでの GET リクエストと HEAD リクエストにレスポンスします。	バケット内のオブジェクトキーのリストを返します。	ウェブサイト設定で指定されているインデックスドキュメントを返します。
Secure Sockets Layer (SSL) のサポート	SSL 接続をサポートします。	SSL 接続をサポートしません。

Amazon S3 エンドポイントの完全なリストについては、AWS 全般のリファレンス の「Amazon S3 エンドポイントとクォータ」を参照してください。

ウェブサイトのホスティングの有効化

バケットを静的ウェブサイトとして設定する場合は、静的ウェブサイトホスティングの有効化、インデックスドキュメントの設定、およびアクセス許可の設定を行う必要があります。

Amazon S3 コンソール、REST API、AWS SDK、AWS CLI、または AWS CloudFormation を使用して、静的ウェブサイトホスティングを有効にすることができます。

カスタムドメインを使用してウェブサイトを設定するには、「チュートリアル: Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定」を参照してください。

S3 コンソールの使用

静的ウェブサイトホスティングを有効にするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケット] リストで、静的ウェブサイトホスティングを有効にするバケットの名前を選択します。
3. [プロパティ] を選択します。
4. [静的ウェブサイトホスティング] で [編集] を選択します。
5. [このバケットを使用してウェブサイトをホストする] を選択します。
6. [静的ウェブサイトホスティング] で [有効化] を選択します。
7. [インデックスドキュメント] に、インデックスドキュメントのファイル名 (通常は `index.html`) を入力します。

   インデックスドキュメント名の大文字と小文字は区別されます。この名前は、S3 バケットにアップロードする HTML インデックスドキュメントのファイル名と正確に一致する必要があります。バケットをウェブサイトホスティング用に設定するときは、インデックスドキュメントを指定する必要があります。Amazon S3 からこのインデックスドキュメントが返されるのは、ルートドメインまたはサブフォルダに対するリクエストが行われたときです。詳細については、「インデックスドキュメントの設定」を参照してください。
8. 4XX クラスエラーに関する独自のカスタムエラードキュメントを指定する場合は、[エラードキュメント] にカスタムエラードキュメントのファイル名を入力します。

   エラードキュメント名の大文字と小文字は区別されます。この名前は、S3 バケットにアップロードする HTML エラードキュメントのファイル名と正確に一致する必要があります。カスタムエラードキュメントを指定しない場合、エラーが発生すると、Amazon S3 からデフォルトの HTML エラードキュメントが返されます。詳細については、「カスタムエラードキュメントの設定」を参照してください。
9. (オプション) 高度なリダイレクトツールを指定する場合は、[Rredirection rules] (リダイレクトルール) に、JSON を入力してルールを記述します。

例えば、条件に応じてリクエストのルーティング先を変えることができます。この条件として使用できるのは、リクエストの中の特定のオブジェクトキー名またはプレフィックスです。詳細については、「高度な条件付きリダイレクトを使用するようにリダイレクトルールを設定する」を参照してください。

10. [Save changes] (変更の保存) をクリックします。

    Amazon S3 では、バケットの静的ウェブサイトホスティングを有効にします。ページの下部の [静的ウェブサイトホスティング] の下に、バケットのウェブサイトエンドポイントが表示されます。

11. [静的 ウェブサイトホスティング] の下のエンドポイントを書き留めます。

    [Endpoint (エンドポイント)] は、バケットの Amazon S3 ウェブサイトエンドポイントです。バケットを静的ウェブサイトとして設定すると、このエンドポイントを使用してウェブサイトをテストできます。

REST API の使用

REST リクエストを直接送信して静的ウェブサイトホスティングを有効にする方法の詳細については、Amazon Simple Storage Service API リファレンスの以下のセクションを参照してください。

- PUT Bucket website
- GET Bucket website
- DELETE Bucket website

AWS SDK の使用

Amazon S3 で静的ウェブサイトをホスティングするには、Amazon S3 バケットをウェブサイトホスティング用に設定してから、ウェブサイトのコンテンツをそのバケットにアップロードします。AWS SDK を使用して、プログラムでウェブサイト設定を作成、更新、削除することもできます。SDK は、Amazon S3 REST API のラッパークラスを提供します。アプリケーションの必要性に応じて、アプリケーションから直接 REST API リクエストを送信することができます。

.NET

次の例では、AWS SDK for .NET を使用してバケットのウェブサイト設定を管理する方法を示します。バケットにウェブサイト設定を追加するには、バケット名とウェブサイト設定を指定します。ウェブサイト設定にはインデックスドキュメントを含める必要があります。また、オプショ

ンとしてエラードキュメントを含めることができます。これらのドキュメントは、バケット内に保存済みであることが必要です。詳細については、「PUT Bucket ウェブサイト」を参照してください。Amazon S3 ウェブサイト機能の詳細については、Amazon S3 を使用して静的ウェブサイトをホスティングする を参照してください。

次の C# コード例は、指定したバケットにウェブサイト設定を追加しています。この設定で、インデックスドキュメント名とエラードキュメント名の両方を指定します。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class WebsiteConfigTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string indexDocumentSuffix = "*** index object key ***"; //
 For example, index.html.
        private const string errorDocument = "*** error object key ***"; // For
 example, error.html.
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddWebsiteConfigurationAsync(bucketName, indexDocumentSuffix,
 errorDocument).Wait();
        }

        static async Task AddWebsiteConfigurationAsync(string bucketName,
                                            string indexDocumentSuffix,
                                            string errorDocument)
        {
            try
            {
                // 1. Put the website configuration.
```

```
                PutBucketWebsiteRequest putRequest = new PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument
                    }
                };
                PutBucketWebsiteResponse response = await
 client.PutBucketWebsiteAsync(putRequest);

                // 2. Get the website configuration.
                GetBucketWebsiteRequest getRequest = new GetBucketWebsiteRequest()
                {
                    BucketName = bucketName
                };
                GetBucketWebsiteResponse getResponse = await
 client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine("Index document: {0}",
 getResponse.WebsiteConfiguration.IndexDocumentSuffix);
                Console.WriteLine("Error document: {0}",
 getResponse.WebsiteConfiguration.ErrorDocument);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

次の PHP の例では、指定したバケットにウェブサイト設定を追加します。create_website_config メソッドで、インデックスドキュメント名とエラードキュメント名を明示的に指定します。さらに、ウェブサイト設定を取得し、レスポンスを出力しま

す。Amazon S3 ウェブサイト機能の詳細については、Amazon S3 を使用して静的ウェブサイトをホスティングする を参照してください。

AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);


// Add the website configuration.
$s3->putBucketWebsite([
    'Bucket'                => $bucket,
    'WebsiteConfiguration'  => [
        'IndexDocument' => ['Suffix' => 'index.html'],
        'ErrorDocument' => ['Key' => 'error.html']
    ]
]);

// Retrieve the website configuration.
$result = $s3->getBucketWebsite([
    'Bucket' => $bucket
]);
echo $result->getPath('IndexDocument/Suffix');

// Delete the website configuration.
$s3->deleteBucketWebsite([
    'Bucket' => $bucket
]);
```

AWS CLI の使用

AWS CLI を使用して S3 バケットを静的ウェブサイトとして設定する方法の詳細については、AWS CLI コマンドリファレンスのウェブサイトを参照してください。

次に、インデックスドキュメントの設定とアクセス許可の設定を行う必要があります。詳細については、「インデックスドキュメントの設定」および「ウェブサイトアクセスのアクセス許可の設定」を参照してください。

オプションとして、エラードキュメント、ウェブトラフィックのログ記録、またはリダイレクトを設定することもできます。

インデックスドキュメントの設定

ウェブサイトホスティングを有効にする場合は、インデックスドキュメントの設定とアップロードも必要です。インデックスドキュメントは、リクエストがウェブサイトのルートまたはサブフォルダに対して行われた場合に Amazon S3 によって返されるウェブページです。たとえば、ユーザーがブラウザに `http://www.example.com` と入力した場合は、このユーザーは特定のページをリクエストしてはいません。この場合、Amazon S3 は、インデックスドキュメントを提供し、これがデフォルトページと見なされることがあります。

バケットに対して静的ウェブサイトホスティングを有効にする場合は、インデックスドキュメントの名前 (`index.html` など) を入力します。バケットに対して静的ウェブサイトホスティングを有効にした後、インデックスドキュメント名を含む HTML ファイルをバケットにアップロードします。

ルートレベル URL の末尾のスラッシュは省略可能です。たとえば、`index.html` をウェブサイトのインデックスドキュメントとして設定した場合は、以下の URL のどちらからも `index.html` が返されます。

```
http://example-bucket.s3-website.Region.amazonaws.com/
http://example-bucket.s3-website.Region.amazonaws.com
```

Amazon S3 ウェブサイトのエンドポイントの詳細については、「ウェブサイトエンドポイント」を参照してください。

インデックスドキュメントとフォルダ

Amazon S3 では、バケットはオブジェクトのフラットコンテナです。つまり、コンピュータのファイルシステムとは異なり、階層構造ではありません。ただし、論理的な階層を作成するには、フォルダ構造を暗示させる名前をオブジェクトキーに付けます。

たとえば、以下のキー名を持つ 3 つのオブジェクトを含むバケットがあるとします。これらは、物理的な階層構造で格納されているわけではありませんが、キー名から次のような論理フォルダ構造を推測できます。

- sample1.jpg — オブジェクトはバケットのルートにあります。
- photos/2006/Jan/sample2.jpg — オブジェクトは photos/2006/Jan サブフォルダにあります。
- photos/2006/Feb/sample3.jpg — オブジェクトは photos/2006/Feb サブフォルダにあります。

Amazon S3 コンソールでは、バケットにフォルダを作成することもできます。たとえば、photos という名前のフォルダを作成できます。バケットまたはバケット内の photos フォルダにオブジェクトをアップロードできます。オブジェクト sample.jpg をバケットに追加する場合、キー名は sample.jpg です。オブジェクトを photos フォルダにアップロードする場合、オブジェクトキー名は photos/sample.jpg です。

フォルダ構造をバケット内に作成する場合は、各レベルにインデックスドキュメントが存在している必要があります。各フォルダでは、インデックスドキュメントは同じ名前 (index.html など) であることが必要です。ユーザーが指定した URL がフォルダルックアップに似ている場合は、末尾のスラッシュの有無によって、ウェブサイトエンドポイントの動作が決まります。たとえば、末尾にスラッシュのある次の URL はphotos/index.html インデックスドキュメントを返します。

```
http://bucket-name.s3-website.Region.amazonaws.com/photos/
```

ただし、前の URL から末尾のスラッシュを除外した場合、Amazon S3 はまずバケット内のオブジェクト photos を検索します。photos オブジェクトが見つからない場合、インデックスドキュメント photos/index.html が検索されます。見つかった場合、Amazon S3 は 302 Found メッセージを返し、photos/ キーを指し示します。それ以降の photos/ に対するリクエストについて、Amazon S3 は photos/index.html を返します。インデックスドキュメントも見つからない場合、Amazon S3 はエラーを返します。

インデックスドキュメントを設定する

S3 コンソールを使用してインデックスドキュメントを設定するには、次の手順に従います。REST API、AWS SDK、AWS CLI、または AWS CloudFormation を使用してインデックスドキュメントを設定することもできます。

> **Note**
>
> バージョニングが有効なバケットでは、index.html のコピーを複数アップロードできますが、解決されるのは最新バージョンのみです。S3 バージョニングの詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

バケットに対して静的ウェブサイトホスティングを有効にする場合は、インデックスドキュメントの名前 (**index.html** など) を入力します。バケットに対して静的ウェブサイトホスティングを有効にした後、インデックスドキュメント名を含む HTML ファイルをバケットにアップロードします。

インデックスドキュメントを設定するには

1. index.html ファイルを作成します。

   index.html ファイルがない場合は、以下の HTML を使用して作成できます。

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" >
   <head>
       <title>My Website Home Page</title>
   </head>
   <body>
     <h1>Welcome to my website</h1>
     <p>Now hosted on Amazon S3!</p>
   </body>
   </html>
   ```

2. インデックスファイルをローカルに保存します。

   インデックスドキュメントファイル名は、[静的ウェブサイトホスティング] ダイアログボックスで入力したインデックスドキュメント名と正確に一致する必要があります。インデックスドキュメント名では、大文字と小文字が区別されます。例えば、[静的ウェブサイトホスティング] ダイアログボックスの [インデックスドキュメント] 名に「index.html」と入力する場合、インデックスドキュメントファイル名も index.html ではなく Index.html である必要があります。

3. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

4. [バケット] リストで、静的ウェブサイトホスティングに使用するバケットの名前を選択します。

5. バケットに対して静的ウェブサイトホスティングを有効にし、インデックスドキュメントの正確な名前 (index.html など) を入力します。詳細については、「ウェブサイトのホスティングの有効化」を参照してください。

   静的ウェブサイトホスティングを有効にしたら、ステップ 6 に進みます。
6. インデックスドキュメントをバケットにアップロードするには、以下のいずれかを実行します。

   - インデックスファイルをコンソールバケットのリストにドラッグアンドドロップします。
   - [Upload] を選択し、プロンプトに従ってインデックスファイルを選択してアップロードします。

   手順については、「オブジェクトのアップロード」を参照してください。
7. (オプション) 他のウェブサイトコンテンツをバケットにアップロードします。

次に、ウェブサイトへのアクセス許可を設定する必要があります。詳細については、「ウェブサイトアクセスのアクセス許可の設定」を参照してください。

オプションとして、エラードキュメント、ウェブトラフィックのログ記録、またはリダイレクトを設定することもできます。

カスタムエラードキュメントの設定

バケットを静的ウェブサイトとして設定した後、エラーが発生すると、Amazon S3 から HTML エラードキュメントが返されます。オプションで、エラーが発生したときに Amazon S3 からそのドキュメントが返されるように、カスタムエラードキュメントを含むバケットを設定できます。

> **Note**
>
> ただし、エラーが発生した場合、一部のブラウザは独自のエラーメッセージを表示し、Amazon S3 が返すエラードキュメントを無視することがあります。たとえば、HTTP 404 Not Found エラーが発生した場合、Google Chrome は Amazon S3 が返すエラードキュメントを無視し、独自のエラーを表示することがあります。

トピック

- Amazon S3 HTTP レスポンスコード
- カスタムエラードキュメントの設定

Amazon S3 HTTP レスポンスコード

次の表は、エラーの発生時に Amazon S3 が返す HTTP レスポンスコードのサブセットをまとめたものです。

HTTP エラーコード	説明
301 Moved Permanently	ユーザーが Amazon S3 ウェブサイトエンドポイント (`http://s3-website.` *`Region`*`.amazonaws.com/`) に直接リクエストを送信した場合、Amazon S3 は 301 Moved Permanently レスポンスを返し、それらのリクエストを `https://aws.amazon.com/s3/` にリダイレクトします。
[302 Found]	Amazon S3 が末尾にスラッシュのないキー `x`、`http://`*`bucket-name`*`.s3-website.` *`Region`*`.amazonaws.com/x` に対するリクエストを受け取った場合、まずキー名 `x` のオブジェクトを検索します。オブジェクトが見つからない場合、Amazon S3 はリクエストがサブフォルダ `x` に対するものであると判断し、末尾にスラッシュを追加して、リクエストをリダイレクトし、302 Found を返します。
[304 Not Modified]	Amazon S3 は、リクエストヘッダー `If-Modified-Since` 、`If-Unmodified-Since` 、`If-Match`、および/または `If-None-Match` を使用して、リクエストされたオブジェクトがクライアントで保持されているキャッシュされたコピーと同じであるかどうかを判断します。オブジェクトが同じである場合、ウェブサイトエンドポイントは [304 Not Modified] レスポンスを返します。
[400 Malformed Request]	ユーザーが誤ったリージョンエンドポイント経由でバケットにアクセスを試みると、ウェブサイトエンドポイントは [400 Malformed Request] で応答します。
403 Forbidden	ユーザーのリクエストが公開で読み取ることのできないオブジェクトに変換する場合、ウェブサイトエンドポイントは [403 Forbidden] で応答します。オブジェクト所有者は、バケットポリシーまたは ACL を使用して、オブジェクトを公開する必要があります。

HTTP エラーコード	説明
404 Not Found	ウェブサイトエンドポイントは、以下の理由により [404 Not Found] で応答します。 • ウェブサイトの URL が存在しないオブジェクトキーを参照しているものと、Amazon S3 が判断した。 • リクエストが存在しないインデックスドキュメントに対するものであると、Amazon S3 が推論した。 • URL に指定されたバケットが存在しない。 • URL に指定されたバケットは存在するが、ウェブサイトとして設定されていない。 [404 Not Found] で返すドキュメントをカスタマイズすることができます。必ず、このドキュメントをウェブサイトとして設定されたバケットにアップロードすると共に、このドキュメントを使用するようにウェブサイトホスティング設定を行ってください。 Amazon S3 がオブジェクトまたはインデックスドキュメントのリクエストとして URL をどのように解釈するかについては、「インデックスドキュメントの設定」を参照してください。
[500 Service Error]	内部サーバーエラーが発生した場合、ウェブサイトエンドポイントは [500 Service Error] で応答します。
503 Service Unavailable	Amazon S3 がユーザーのリクエスト頻度を減らす必要があると判断した場合に、ウェブサイトエンドポイントは 503 Service Unavailable で応答します。

これらの各エラーに対して、Amazon S3 は定義済みの HTML メッセージを返します。[403 Forbidden] レスポンスで返される HTML メッセージの例を次に示します。

カスタムエラードキュメントの設定

バケットを静的ウェブサイトとして設定する場合、ユーザーフレンドリーなエラーメッセージと追加のヘルプを含むカスタムエラードキュメントを提供できます。Amazon S3 は HTTP 4XX クラスのエラーコードに対してのみ、カスタムエラードキュメントを返します。

S3 コンソールを使用してカスタムエラードキュメントを設定するには、次の手順に従います。REST API、AWS SDK、AWS CLI、または AWS CloudFormation を使用してエラードキュメントを設定することもできます。詳細については、以下を参照してください。

- Amazon Simple Storage Service API リファレンスの「PutBucketWebsite」
- AWS CloudFormation ユーザーガイドの AWS::S3::Bucket WebsiteConfiguration
- AWS CLI コマンドリファレンスの put-bucket-website

バケットに対して静的ウェブサイトホスティングを有効にするときは、エラードキュメントの名前 (例: **`404.html`**) を入力します。バケットに対して静的ウェブサイトホスティングを有効にしたら、エラードキュメント名を含む HTML ファイルをバケットにアップロードします。

エラードキュメントを設定するには

1. エラードキュメント (例: `404.html`) を作成します。
2. エラードキュメントのファイルをローカルに保存します。

エラードキュメントの名前は、大文字と小文字を区別し、静的ウェブサイトホスティングを有効にする際に入力した名前と厳密に一致している必要があります。たとえば、[Static website hosting] (静的ウェブサイトホスティング) ダイアログボックスの [Error document] (エラードキュメント) 名に `404.html` と入力する場合、エラードキュメントのファイル名も `404.html` である必要があります。

3. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
4. [バケット] リストで、静的ウェブサイトホスティングに使用するバケットの名前を選択します。
5. バケットに対して静的ウェブサイトホスティングを有効にし、エラードキュメントの正確な名前 (例: `404.html`) を入力します。詳細については、ウェブサイトのホスティングの有効化およびカスタムエラードキュメントの設定を参照してください。

   静的ウェブサイトホスティングを有効にしたら、ステップ 6 に進みます。
6. エラードキュメントをバケットにアップロードするには、以下のいずれかを実行します。

   - エラードキュメントファイルをコンソールバケットのリストにドラッグアンドドロップします。
   - [Upload] を選択し、プロンプトに従ってインデックスファイルを選択してアップロードします。

   手順については、「オブジェクトのアップロード」を参照してください。

ウェブサイトアクセスのアクセス許可の設定

バケットを静的ウェブサイトとして設定する場合、このウェブサイトを公開するときは、パブリック読み取りアクセスを許可できます。バケットをパブリックに読み取り可能にするには、バケットのパブリックアクセスブロック設定を無効にし、パブリック読み取りアクセスを許可するバケットポリシーを記述する必要があります。バケットにバケット所有者が所有していないオブジェクトが含まれている場合は、すべてのユーザーに読み取りアクセスを許可するオブジェクトアクセスコントロールリスト (ACL) の追加が必要になる場合があります。

バケットのパブリックアクセスのブロック設定を無効にするのではなく、ウェブサイトを公開したい場合は、Amazon CloudFront ディストリビューションを作成して、静的ウェブサイトを提供できます。詳細については、Amazon CloudFront によるウェブサイトの高速化 および「Amazon Route 53

デベロッパーガイド」の「Amazon CloudFront ディストリビューションを使用して静的ウェブサイトを提供する」を参照してください。

> **Note**
>
> ウェブサイトエンドポイント上に、ユーザーがリクエストしたオブジェクトが存在しない場合は、Amazon S3 から HTTP レスポンスコード 404 (Not Found) が返されます。オブジェクトが存在するが、そのオブジェクトの読み取りアクセス許可が付与されていない場合は、ウェブサイトエンドポイントから HTTP レスポンスコード 403 (Access Denied) が返されます。ユーザーはこのレスポンスコードを見て、特定のオブジェクトが存在するかどうかを推測できます。この動作を希望しない場合は、バケットのウェブサイトのサポートを有効にしないでください。

トピック

ステップ 1: S3 のパブリックアクセスのブロック設定を編集する

既存のバケットをパブリックアクセスを持つ静的ウェブサイトとして設定する場合は、そのバケットのパブリックアクセスのブロック設定を編集する必要があります。また、アカウントレベルでパブリックアクセスのブロック設定を編集する必要がある場合もあります。Amazon S3 は、アクセスポイントレベル、バケットレベル、アカウントレベルのパブリックアクセスのブロック設定で最も制限の厳しい組み合わせを適用します。

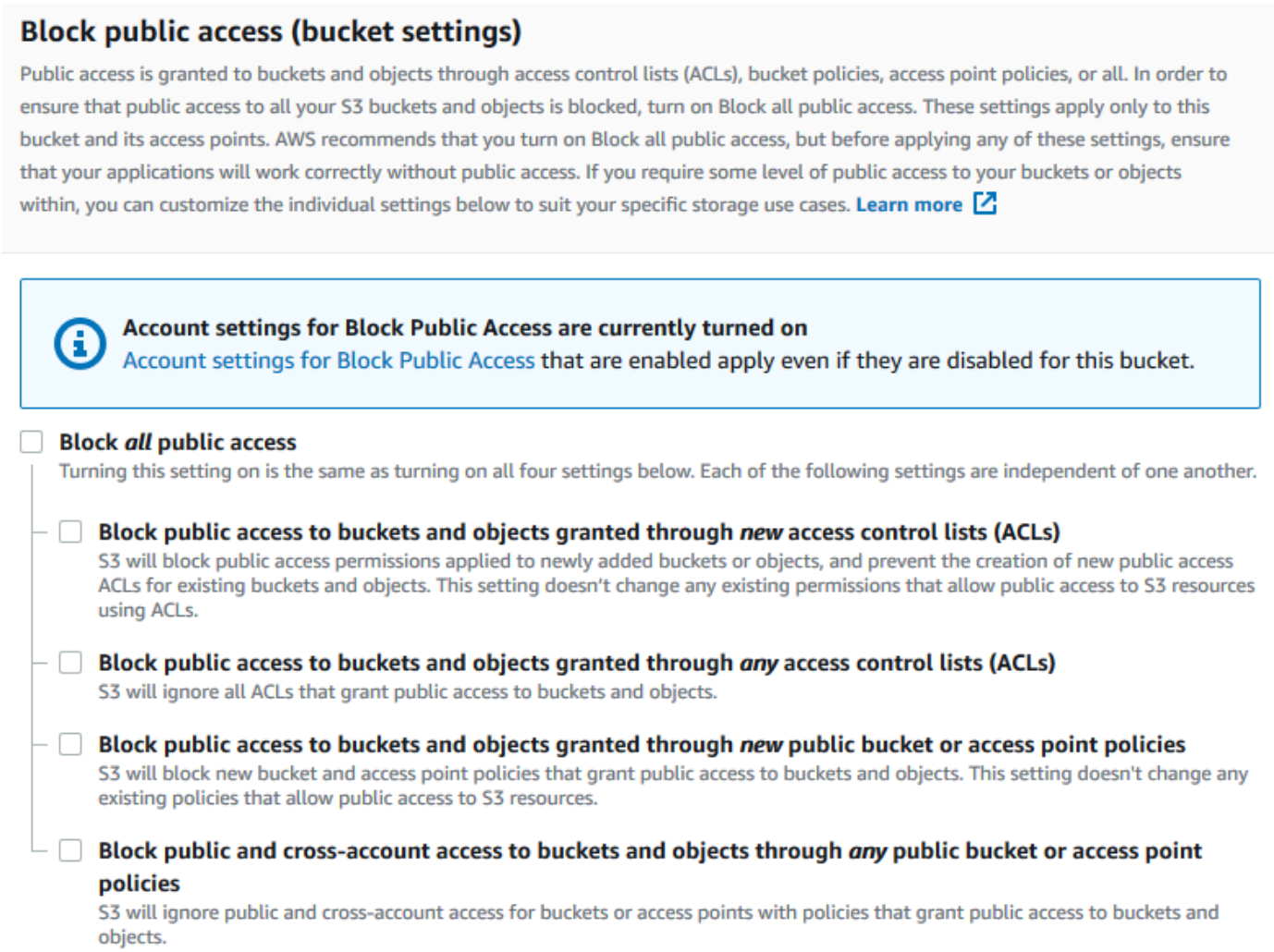
例えば、バケットのパブリックアクセスを許可し、アカウントレベルですべてのパブリックアクセスをブロックした場合、Amazon S3 はバケットへのパブリックアクセスを引き続きブロックします。このシナリオでは、バケットレベルとアカウントレベルのパブリックアクセスのブロック設定を編集する必要があります。詳細については、「Amazon S3 ストレージへのパブリックアクセスのブロック」を参照してください。

デフォルトでは、Amazon S3 はアカウントとバケットへのパブリックアクセスをブロックします。バケットを使用して静的ウェブサイトをホストする場合は、以下のステップを使用して、パブリックアクセスブロック設定を編集できます。

> **Warning**
>
> このステップを完了する前に「Amazon S3 ストレージへのパブリックアクセスのブロック」を読んで、パブリックアクセスを許可することに伴うリスクを理解し、了承します。パブリックアクセスブロック設定をオフにしてバケットをパブリックにすると、インターネット上のだれでもバケットにアクセスできるようになります。バケットへのすべてのパブリックアクセスをブロックすることをお勧めします。

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 静的ウェブサイトとして設定されたバケットの名前を選択します。
3. [Permissions (アクセス許可)] を選択します。
4. [ブロックパブリックアクセス (バケット設定)] で [編集] を選択します。
5. [Block all public access (すべてのパブリックアクセスをブロックする)] をクリアし、[Save changes (変更の保存)] を選択します。

> **Warning**
>
> このステップを完了する前に、「Amazon S3 ストレージへのパブリックアクセスのブロック」を確認し、パブリックアクセスの許可に伴うリスクを理解したうえで了承してください。パブリックアクセスブロック設定をオフにしてバケットをパブリックにすると、インターネット上のだれでもバケットにアクセスできるようになります。バケットへのすべてのパブリックアクセスをブロックすることをお勧めします。



Amazon S3 は、バケットのパブリックアクセスブロック設定をオフにします。パブリックで静的ウェブサイトを作成するには、バケットポリシーを追加する前に、アカウントのブロックパブリックアクセス設定を編集する必要があります。パブリックアクセスのブロックのアカウント設定が現在有効になっている場合は、[Block public access (bucket settings) (パブリックアクセスのブロック (バケット設定))] の下にメモが表示されます。

ステップ 2: バケットポリシーを追加する

バケット内のオブジェクトをパブリックに読み取り可能にするには、すべてのユーザーに `s3:GetObject` アクセス許可を付与するバケットポリシーを作成する必要があります。

S3 のパブリックアクセスブロック設定を編集した後で、バケットへのパブリック読み取りアクセスを許可するバケットポリシーを追加できます。パブリック読み取りアクセスを許可すると、インターネット上のだれでもバケットにアクセスできるようになります。

> **Important**
>
> 次のポリシーは、単なる例として、バケットのコンテンツへのフルアクセスを許可します。このステップに進む前に、「Amazon S3 バケット内のファイルを保護するにはどうすればよいですか?」を確認して、S3 バケット内のファイルを保護するためのベストプラクティスと、パブリックアクセスの許可に伴うリスクを理解してください。

1. [バケット] で、バケットの名前を選択します。
2. [Permissions (アクセス許可)] を選択します。
3. [Bucket Policy (バケットポリシー)] で [編集] を選択します。
4. ウェブサイトのパブリック読み取りアクセスを許可するには、次のバケットポリシーをコピーし、[Bucket policy editor (バケットポリシーエディター)] に貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. `Resource` をバケット名に更新します。

   前述のバケットポリシーの例では、*`Bucket-Name`* はバケット名のプレースホルダーです。このバケットポリシーを独自のバケットで使用するには、バケット名に合わせてこの名前を更新する必要があります。

6. [Save changes] (変更の保存) をクリックします。

   バケットポリシーが正常に追加されたことを示すメッセージが表示されます。

   `Policy has invalid resource` というエラーが表示された場合は、バケットポリシー内のバケット名がバケット名と一致していることを確認します。バケットポリシーの追加については、「S3 バケットポリシーを追加する方法」を参照してください。

   エラーメッセージが表示され、バケットポリシーを保存できない場合は、アカウントとバケットの [パブリックアクセスをブロックする] 設定をチェックして、バケットへのパブリックアクセスを許可していることを確認します。

オブジェクトアクセスコントロールリスト

バケットポリシーを使用して、オブジェクトに対するアクセス許可を付与できます。ただし、バケットポリシーは、バケット所有者が所有するオブジェクトにのみ適用されます。バケットに、バケット所有者として所有していないオブジェクトが含まれる場合は、バケット所有者は、オブジェクトのアクセスコントロールリスト (ACL) を使用して、それらのオブジェクトへのパブリック読み取りアクセス許可を付与する必要があります。

S3 オブジェクト所有権は、Amazon S3 バケットレベルの設定で、バケットにアップロードされる新しいオブジェクト所有権を制御し、ACL を無効にするのに使用できます。デフォルトでは、オブジェクト所有権はバケット所有者の強制設定に設定され、すべての ACL は無効になります。ACL を無効にすると、バケット所有者はバケット内のすべてのオブジェクトを所有し、アクセス管理ポリシーのみを使用してデータへのアクセスを管理します。

Amazon S3 の最新のユースケースの大部分では ACL を使用する必要がなくなっています。オブジェクトごとに個別に制御する必要がある通常ではない状況を除き、ACL は無効にしておくことをお勧めします。ACL を無効にすると、誰がオブジェクトをバケットにアップロードしたかに関係なく、ポリシーを使用してバケット内のすべてのオブジェクトへのアクセスを制御できます。詳細については、「オブジェクトの所有権の制御とバケットの ACL の無効化。」を参照してください。

> **Important**
>
> バケットが S3 オブジェクト所有権のバケット所有者強制設定を使用している場合、ポリシーを使用してバケットとバケット中のオブジェクトへのアクセスを許可する必要があります。バケット所有者強制設定が有効になっている場合、アクセスコントロールリスト (ACL)

の設定または ACL の更新は失敗し、`AccessControlListNotSupported` エラーコードが返されます。ACL の読み取りリクエストは引き続きサポートされています。

ACL を使用してオブジェクトを公開で読み取り可能にするには、次の許可要素に示すように、`AllUsers` グループに READ アクセス許可を付与します。この許可要素をオブジェクト ACL に追加します。ACL の管理の詳細については、「アクセスコントロールリスト (ACL) の概要」を参照してください。

```
<Grant>
  <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
```

(オプション) ウェブトラフィックのログ記録

オプションで、静的ウェブサイトとして設定されているバケットで Amazon S3 サーバーアクセスログを有効にできます。サーバーアクセスのログ記録では、バケットに対するリクエストの詳細が記録されます。詳細については、「サーバーアクセスログによるリクエストのログ記録」を参照してください。Amazon CloudFront を使用してウェブサイトを高速化する場合は、CloudFront のログ記録を使用することもできます。詳細については、Amazon CloudFront デベロッパーガイドの「アクセスログの設定および使用」を参照してください。

静的ウェブサイトバケット用のサーバーアクセスのログ記録を有効にするには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 静的ウェブサイトとして設定しているバケットを作成したのと同じリージョンで、ログ記録用のバケットを作成します (例: `logs.example.com`)。
3. サーバアクセスのログを記録するログファイル用のフォルダを作成します (例: `logs`)。
4. (オプション) CloudFront を使用してウェブサイトのパフォーマンスを向上させる場合は、CloudFront ログファイル用のフォルダを作成します (例: `cdn`)。

   詳細については、「Amazon CloudFront によるウェブサイトの高速化」を参照してください。
5. [Buckets (バケット)] リストで、バケット名を選択します。

6. [プロパティ] を選択します。
7. [Server access logging (サーバーアクセスのログ記録)] で、[Edit (編集)] を選択します。
8. [Enable] を選択します。
9. [Target bucket (ターゲットバケット)] で、サーバーアクセスログのバケットとフォルダの宛先を選択します。

   - フォルダとバケットの場所を参照します。
     1. [Browse S3(S3 の参照)] を選択します。
     2. バケット名を選択し、ログフォルダを選択します。
     3. [Choose path (パスの選択)] を選択します。
   - S3 バケットパスを入力します (例: **`s3://logs.example.com/logs/`**)。
10. [Save changes] (変更の保存) をクリックします。

    ログバケットで、ログにアクセスできるようになりました。Amazon S3 は 2 時間おきにログバケットにウェブサイトアクセスログを書き込みます。

(オプション) ウェブページリダイレクトの設定

Amazon S3 バケットに対して静的ウェブサイトホスティングが設定されている場合は、バケットまたはそのバケット内のオブジェクトに対してリダイレクトを設定できます。リダイレクトを設定するには、次のオプションがあります。

トピック

- バケットのウェブサイトエンドポイントに対するリクエストを別のバケットまたはドメインにリダイレクトする
- 高度な条件付きリダイレクトを使用するようにリダイレクトルールを設定する
- オブジェクトのリクエストをリダイレクトする

バケットのウェブサイトエンドポイントに対するリクエストを別のバケットまたはドメインにリダイレクトする

バケットのウェブサイトエンドポイントに対するすべてのリクエストを、別のバケットまたはドメインにリダイレクトできます。すべてのリクエストをリダイレクトする場合、ウェブサイトエンドポイ

ントに対して行われたすべてのリクエストが指定されたバケットまたはドメインにリダイレクトされます。

例えば、ルートドメインが example.com であり、http://example.com および http://www.example.com の両方のためのリクエストに対応する場合は、example.com および www.example.com という 2 つのバケットを作成する必要があります。次に、example.com バケット内のコンテンツを保持し、他の www.example.com バケットからすべてのリクエストを example.com バケットにリダイレクトするように設定します。詳細については、「カスタムドメイン名を使用して静的ウェブサイトを設定する」を参照してください。

バケットウェブサイトエンドポイントにリクエストをリダイレクトするには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. [Buckets (バケット)] で、リクエストをリダイレクトするバケットの名前 (www.example.com など)を選択します。
3. [プロパティ] を選択します。
4. [静的ウェブサイトホスティング] で [編集] を選択します。
5. [Redirect requests for an object (オブジェクトのリクエストをリダイレクト)] を選択します。
6. [Host name (ホスト名)] ボックスに、バケットまたはカスタムドメインのウェブサイトエンドポイントを入力します。

   たとえば、ルートドメインアドレスにリダイレクトする場合は、「**example.com**」と入力します。
7. [Protocol (プロトコル)] で、リダイレクトされた要求のプロトコル ([none]、[http]、または [https]) を選択します。

   プロトコルを指定しない場合、デフォルトのオプションは [none] です。
8. [Save changes] (変更の保存) をクリックします。

高度な条件付きリダイレクトを使用するようにリダイレクトルールを設定する

高度な条件付きリダイレクトルールを使用すると、条件に応じてリクエストのルーティング先を変えることができます。この条件として使用できるのは、特定のオブジェクトキー名、リクエストの中のプレフィックス、またはレスポンスコードです。たとえば、バケット内のあるオブジェクトを削除する、または名前を変更する場合は、リクエストを別のオブジェクトにリダイレクトするルーティング

ルールを追加します。フォルダを使用できなくする場合は、ルーティングルールを追加して別のウェブページにリクエストをリダイレクトできます。ルーティングルールは、エラー状態を処理するために追加することもできます。その場合は、エラーの処理時に、エラーを発生させたリクエストを別のドメインにリダイレクトします。

バケットに対して静的ウェブサイトホスティングを有効にすると、オプションで高度なリダイレクトルールを指定できます。Amazon S3 では、ウェブサイト設定ごとのルーティングルールの数が 50 に制限されています。50 を超えるルーティングルールが必要な場合は、オブジェクトリダイレクトを使用できます。詳細については、「S3 コンソールの使用」を参照してください。

REST API を使用したルーティングルールの設定の詳細については、Amazon Simple Storage Service API リファレンスの「PutBucketWebsite」を参照してください。

> **Important**
>
> 新しい Amazon S3 コンソールでリダイレクトルールを作成するには、JSON を使用する必要があります。JSON の例については、「リダイレクトルールの例」を参照してください。

静的ウェブサイトのリダイレクトルールを設定するには

静的ウェブサイトホスティングがすでに有効になっているバケットのリダイレクトルールを追加するには、次の手順に従います。

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. [バケット] リストで、静的ウェブサイトとして設定したバケットの名前を選択します。
3. [プロパティ] を選択します。
4. [静的ウェブサイトホスティング] で [編集] を選択します。
5. [Redirection rules (リダイレクトルール)] ボックスに、JSON にリダイレクトルールを入力します。

   S3 コンソールでは、JSON を使用してルールを記述します。JSON の例については、「リダイレクトルールの例」を参照してください。Amazon S3 では、ウェブサイト設定ごとのルーティングルールの数が 50 に制限されています。
6. [Save changes] (変更の保存) をクリックします。

ルーティングルール要素

JSON および XML でウェブサイト設定のルーティングルールを定義するための一般的な構文を次に示します。新しい S3 コンソールでリダイレクトルールを設定するには、JSON を使用する必要があります。JSON の例については、「リダイレクトルールの例」を参照してください。

JSON

```
[
    {
      "Condition": {
        "HttpErrorCodeReturnedEquals": "string",
        "KeyPrefixEquals": "string"
      },
      "Redirect": {
        "HostName": "string",
        "HttpRedirectCode": "string",
        "Protocol": "http"|"https",
        "ReplaceKeyPrefixWith": "string",
        "ReplaceKeyWith": "string"
      }
    }
  ]

Note: Redirect must each have at least one child element. You can have either
 ReplaceKeyPrefix with or ReplaceKeyWith but not both.
```

XML

```
<RoutingRules> =
    <RoutingRules>
         <RoutingRule>...</RoutingRule>
         [<RoutingRule>...</RoutingRule>
         ...]
    </RoutingRules>

<RoutingRule> =
   <RoutingRule>
      [ <Condition>...</Condition> ]
      <Redirect>...</Redirect>
   </RoutingRule>

<Condition> =
```

```
   <Condition>
      [ <KeyPrefixEquals>...</KeyPrefixEquals> ]
      [ <HttpErrorCodeReturnedEquals>...</HttpErrorCodeReturnedEquals> ]
   </Condition>
    Note: <Condition> must have at least one child element.

<Redirect> =
   <Redirect>
      [ <HostName>...</HostName> ]
      [ <Protocol>...</Protocol> ]
      [ <ReplaceKeyPrefixWith>...</ReplaceKeyPrefixWith>  ]
      [ <ReplaceKeyWith>...</ReplaceKeyWith> ]
      [ <HttpRedirectCode>...</HttpRedirectCode> ]
   </Redirect>

Note: <Redirect> must have at least one child element. You can have either
 ReplaceKeyPrefix with or ReplaceKeyWith but not both.
```

次の表は、ルーティングルールの要素をまとめたものです。

名前	説明
RoutingRules	RoutingRule 要素のコレクションを入れるコンテナ。
RoutingRule	条件とリダイレクトを明示したルール。この条件が満たされたときにこのリダイレクトが適用されます。 条件: • RoutingRules コンテナには、少なくとも 1 つのルーティングルールを含める必要があります。
Condition	条件を記述するためのコンテナ。この条件が満たされているときに、指定されたリダイレクトが適用されます。ルーティングルールの中に条件がない場合は、そのルールはすべてのリクエストに適用されます。

名前	説明
KeyPrefixEquals	リクエストをリダイレクトするオブジェクトキー名のプレフィックス。 KeyPrefixEquals HttpErrorCodeReturnedEquals が指定されていない場合は、 は必須です。KeyPrefixEquals と HttpErrorCodeReturnedEquals の両方が指定されている場合は、両方が真である場合に、条件が満たされていると見なされます。
HttpErrorCodeReturnedEquals	この HTTP エラーコードに一致している場合にのみ、リダイレクトが適用されます。エラーが発生した場合、エラーコードがこの値に等しい場合は、指定されたリダイレクトが適用されます。 HttpErrorCodeReturnedEquals KeyPrefixEquals が指定されていない場合は、 は必須です。KeyPrefixEquals と HttpErrorCodeReturnedEquals の両方が指定されている場合は、両方が真である場合に、条件が満たされていると見なされます。
Redirect	コンテナ要素。リクエストのリダイレクトに関する指示を表すのに使用されます。リクエストを別のホストや別のページにリダイレクトすることも、別のプロトコルの使用を指定することもできます。RoutingRule には Redirect 要素が必要です。1 つの Redirect 要素の中に、Protocol、HostName、ReplaceKeyPrefixWith 、ReplaceKeyWith 、HttpRedirectCode の兄弟要素のうち 1 つ以上が存在する必要があります。
Protocol	レスポンスとして返される http ヘッダーで使用されるプロトコル (https または Location) 。 同レベルの要素の 1 つが指定されている場合、Protocol は省略可能です。

名前	説明
HostName	レスポンスとして返される Location ヘッダーで使用されるホスト名。 同レベルの要素の 1 つが指定されている場合、HostName は省略可能です。
ReplaceKeyPrefixWith	リダイレクトリクエストの中の KeyPrefixEquals の値を置き換えるオブジェクトキー名のプレフィックス。 同レベルの要素の 1 つが指定されている場合、ReplaceKeyPrefixWith は省略可能です。これを指定できるのは、ReplaceKeyWith が指定されていない場合のみです。
ReplaceKeyWith	レスポンスとして返される Location ヘッダーで使用されるオブジェクトキー。 同レベルの要素の 1 つが指定されている場合、ReplaceKeyWith は省略可能です。これを指定できるのは、ReplaceKeyPrefixWith が指定されていない場合のみです。
HttpRedirectCode	レスポンスとして返される Location ヘッダーで使用される HTTP リダイレクトコード。 同レベルの要素の 1 つが指定されている場合、HttpRedirectCode は省略可能です。

リダイレクトルールの例

次の例では、一般的なリダイレクトタスクについて説明します。

> **Important**
>
> 新しい Amazon S3 コンソールでリダイレクトルールを作成するには、JSON を使用する必要があります。

Example 1: キープレフィックスの名前を変更した後にリダイレクトする

バケットに次のオブジェクトが含まれているとします。

- index.html
- docs/article1.html
- docs/article2.html

フォルダ名を docs/ から documents/ に変更することにしました。この変更を行った後は、プレフィックス docs/ に対するリクエストを documents/ にリダイレクトする必要があります。たとえば、docs/article1.html に対するリクエストを documents/article1.html にリダイレクトします。

この場合、次のルーティングルールをウェブサイト設定に追加します。

JSON

```
[
    {
        "Condition": {
            "KeyPrefixEquals": "docs/"
        },
        "Redirect": {
            "ReplaceKeyPrefixWith": "documents/"
        }
    }
]
```

XML

```
  <RoutingRules>
    <RoutingRule>
    <Condition>
```

```
      <KeyPrefixEquals>docs/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyPrefixWith>documents/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

Example 2: 削除されたフォルダに対するリクエストをページにリダイレクトする

images/フォルダ (つまり、キープレフィックスが images/ のすべてのオブジェクト) を削除するとします。ルーティングルールを追加し、その中で、キープレフィックス images/ を持つオブジェクトに対するリクエストをすべて folderdeleted.html というページにリダイレクトするように設定します。

JSON

```
[
    {
        "Condition": {
            "KeyPrefixEquals": "images/"
        },
        "Redirect": {
            "ReplaceKeyWith": "folderdeleted.html"
        }
    }
]
```

XML

```
  <RoutingRules>
    <RoutingRule>
    <Condition>
       <KeyPrefixEquals>images/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyWith>folderdeleted.html</ReplaceKeyWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

Example 3: HTTP エラーの場合にリダイレクトする

リクエストされたオブジェクトが見つからないときに、リクエストを Amazon Elastic Compute Cloud (Amazon EC2) インスタンスにリダイレクトするとします。リダイレクトルールを追加し、その中で、HTTP ステータスコード 404 (Not Found) が返されたときに、リクエストを処理する Amazon EC2 インスタンスにリダイレクトするように設定します。

次に示す例では、オブジェクトキープレフィックス `report-404/` もリダイレクトに挿入しています。たとえば、`ExamplePage.html` というページをリクエストした結果が HTTP 404 エラーとなったときは、指定の Amazon EC2 インスタンスにある `report-404/ExamplePage.html` というページにリクエストをリダイレクトします。ルーティングルールが何も設定されていない場合に HTTP エラー 404 が発生したときは、設定内で指定されているエラードキュメントが返されます。

JSON

```
[
    {
        "Condition": {
            "HttpErrorCodeReturnedEquals": "404"
        },
        "Redirect": {
            "HostName": "ec2-11-22-333-44.compute-1.amazonaws.com",
            "ReplaceKeyPrefixWith": "report-404/"
        }
    }
]
```

XML

```
  <RoutingRules>
    <RoutingRule>
    <Condition>
      <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals >
    </Condition>
    <Redirect>
      <HostName>ec2-11-22-333-44.compute-1.amazonaws.com</HostName>
      <ReplaceKeyPrefixWith>report-404/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

オブジェクトのリクエストをリダイレクトする

オブジェクトのメタデータでウェブサイトのリダイレクト場所を設定することで、オブジェクトのリクエストを別のオブジェクトまたは URL にリダイレクトできます。リダイレクトを設定するには、オブジェクトメタデータに `x-amz-website-redirect-location` プロパティを追加します。Amazon S3 コンソールで、オブジェクトのメタデータ内の [Website Redirect Location (ウェブサイトのリダイレクト場所)] を設定できます。Amazon S3 API を使用する場合は、`x-amz-website-redirect-location` を設定します。ウェブサイトは 301 リダイレクトとしてオブジェクトを解釈します。

リクエストを別のオブジェクトにリダイレクトするには、リダイレクト場所をターゲットオブジェクトのキーに設定します。リクエストを外部 URL にリダイレクトするには、リダイレクト場所を目的の URL に設定します。オブジェクトメタデータの詳細については、システムで定義されたオブジェクトメタデータ を参照してください。

ページリダイレクトを設定するときに、ソースオブジェクトコンテンツを維持することも、削除することもできます。たとえば、バケットに `page1.html` オブジェクトがある場合、このページのリクエストを別のオブジェクト `page2.html` にリダイレクトできます。これには2つのオプションがあります。

- `page1.html` オブジェクトのコンテンツを維持して、ページリクエストをリダイレクトします。
- `page1.html` のコンテンツを削除し、`page1.html` という名前の 0 バイトのオブジェクトをアップロードして、既存のオブジェクトを置き換え、ページリクエストをリダイレクトします。

S3 コンソールの使用

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. [Buckets (バケット)] リストで、静的ウェブサイトとして設定したバケットの名前を選択します (例: `example.com`)。
3. [Objects (オブジェクト)] でオブジェクトを選択します。
4. [Actions (アクション)] を選択し、[Edit metadata (メタデータの編集)] を選択します。
5. [メタデータ] を選択します。
6. [Add Metadata (メタデータの追加)] を選択します。
7. [Type (タイプ)] で、[System Defined (システム定義)] を選択します。
8. [Key (キー)]で、[x-amz-website-redirect-location] を選択します。
9. [値] に、リダイレクト先のオブジェクトのキー名を入力します (例: `/page2.html`)。

同じバケット内の別のオブジェクトでは、値の / プレフィックスが必要です。値を外部 URL に設定することもできます (例: `http://www.example.com`)。

10. [Edit metadata (メタデータの編集)] を選択します。

REST API の使用

次の Amazon S3 API アクションは、リクエストの `x-amz-website-redirect-location` ヘッダーをサポートしています。Amazon S3 はオブジェクトメタデータのヘッダー値を `x-amz-website-redirect-location` として保存します。

- PUT Object
- Initiate Multipart Upload
- POST Object
- PUT Object − Copy

ウェブサイトホスティング用に設定されたバケットには、ウェブサイトエンドポイントと REST エンドポイントの両方があります。リクエストされたページが 301 リダイレクトとして設定されている場合の結果は、リクエストのエンドポイントに応じて、次のようになると考えられます。

- リージョン固有のウェブサイトエンドポイント – Amazon S3 は、`x-amz-website-redirect-location` プロパティの値に従ってページリクエストをリダイレクトします。
- REST エンドポイント – Amazon S3 はページリクエストをリダイレクトしません。リクエストされたオブジェクトを返します。

エンドポイントの詳細については、「ウェブサイトエンドポイントと REST API エンドポイントの主な違い」を参照してください。

ページリダイレクトを設定する場合、オブジェクトのコンテンツを維持するか、削除できます。たとえば、バケットに `page1.html` というオブジェクトがあるとします。

- `page1.html` のコンテンツを維持し、ページリクエストのリダイレクトのみを行う場合は、PUT Object - Copy リクエストを送信して新しい `page1.html` オブジェクトを作成し、送信元として既存の `page1.html` オブジェクトを使用するように設定します。リクエストに、`x-amz-website-redirect-location` ヘッダーを設定します。リクエストが完了すると、元のページのコンテンツは変更されないままですが、Amazon S3 によりページへのすべてのリクエストは指定されたリダイレクト場所にリダイレクトされます。

- page1.html オブジェクトのコンテンツを削除してこのページに対するリクエストをリダイレクトするには、PUT Object リクエストを送信して 0 バイトのオブジェクトをアップロードします。このオブジェクトには、同じオブジェクトキー page1.html を付けます。この PUT リクエストの中で、x-amz-website-redirect-location の page1.html を新しいオブジェクトに設定します。このリクエストが完了すると、page1.html にはコンテンツがない状態になり、リクエストは x-amz-website-redirect-location で指定された場所にリダイレクトされます。

GET Object アクションを使用して、他のオブジェクトメタデータと一緒にオブジェクトを取得すると、Amazon S3 はレスポンスで x-amz-website-redirect-location ヘッダーを返します。

Amazon S3 を使用した開発

このセクションでは、Amazon S3 の使用についてデベロッパーに関連するトピックを説明します。詳細については、以下のトピックを参照してください。

> **Note**
>
> Amazon S3 Express One Zone とディレクトリバケットでマルチパートアップロードを使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

トピック

リクエストの実行

Amazon S3 は REST サービスです。REST API または基礎となる Amazon S3 REST API をラップする AWS SDK (サンプルコードとライブラリを参照) ラッパーライブラリを使用して、Amazon S3 にリクエストを送信できます。これにより、プログラミング作業が簡易になります。

Amazon S3 とのすべてのやり取りは認証されるか匿名で行われます。認証は、アマゾン ウェブ サービス (AWS) 製品にアクセスを試みるリクエスタのアイデンティティを検証するプロセスです。認証リクエストには、リクエストの送信元を認証する署名値を含める必要があります。署名値の一部は、リクエスタの AWS アクセスキー (アクセスキー ID とシークレットアクセスキー) から生成されます。アクセスキーの取得の詳細については、AWS 全般のリファレンス の「セキュリティ認証情報の取得方法」を参照してください。

AWS SDK を使用している場合、指定したキーから、ライブラリによって署名が計算されます。ただし、アプリケーションで直接 REST API を呼び出す場合、署名を計算するコードを作成し、それをリクエストに追加する必要があります。

トピック

アクセスキーについて

以下のセクションでは、認証リクエストの実行に使用できるアクセスキーの種類について説明します。

AWS アカウント アクセスキー

アカウントアクセスキーを使用することで、アカウントが所有する AWS リソースにフルアクセスすることができます。アクセスキーの例を次に示します。

- アクセスキー ID (半角英数字で構成された 20 文字)。例: AKIAIOSFODNN7EXAMPLE
- シークレットアクセスキー (40 文字の文字列)。例: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

アクセスキー ID によって AWS アカウント が一意に識別されます。このアクセスキーを使用して認証リクエストを Amazon S3 に送信できます。

IAM ユーザーアクセスキー

お客様の会社用に 1 つの AWS アカウント を作成することができます。ただし、組織の AWS リソースへのアクセス権を必要とする社員が組織内に複数いる場合があります。AWS アカウント アクセスキーを共有するとセキュリティは低くなり、また、各社員に個別の AWS アカウント を作成することは実用的ではない場合があります。さらに、バケットやオブジェクトなどのリソースは別のアカウントが所有しているので、簡単に共有できません。リソースを共有するには、追加の作業でアクセス許可を付与する必要があります。

この場合、AWS Identity and Access Management (IAM) を使用して、AWS アカウントでユーザーと各自のアクセスキーを作成し、これらのユーザーに該当するリソースへのアクセス許可を付与する IAM ユーザーポリシーをアタッチします。このようなユーザーをより効率的に管理するために、IAM

では、ユーザーのグループを作成し、グループ内のユーザー全員に適用されるグループレベルのアクセス許可を付与できます。

で作成および管理するユーザーは、IAM ユーザーと呼ばれますAWS 親アカウントは、ユーザーが にアクセスできるかどうかを制御しますAWS IAM ユーザーが作成するリソースは、親 AWS アカウント が管理し、支払いを行います。これらの IAM ユーザーは、各自のセキュリティ証明書を使用して Amazon S3 に認証リクエストを送信できます。お客様の AWS アカウント の下でユーザーを作成、管理する方法の詳細については、「AWS Identity and Access Management 製品詳細ページ」を参照してください。

一時的な認証情報

IAM では、IAM ユーザーと各自のアクセスキーを作成できるだけでなく、任意の IAM ユーザーに一時的なセキュリティ認証情報 (一時的なアクセスキーとセキュリティトークン) を付与して、AWS のサービスおよびリソースにアクセスできるようにすることもできます。また、 ではなく、使用しているシステムでユーザーを管理することもできますAWS このようなユーザーはフェデレーティッドユーザーと呼ばれます。また、このユーザーには、AWS リソースにアクセスするために作成したアプリケーションも含まれます。

IAM には、一時的なセキュリティ認証情報をリクエストするための AWS Security Token Service API が用意されています。一時的なセキュリティ認証情報をリクエストするには、AWS STS API または AWS SDK を使用できます。この API は一時的なセキュリティ認証情報 (アクセスキー ID およびシークレットアクセスキー) とセキュリティトークンを返します。これらの認証情報は、リクエスト時に指定した有効期間の間のみ有効です。アクセスキー ID およびシークレットキーの使用方法は、AWS アカウント または IAM ユーザーのアクセスキーを使用してリクエストを送信するときの使用方法と同様です。また、Amazon S3 に送信する各リクエストにトークンを含める必要があります。

IAM ユーザーは、これらの一時的なセキュリティ認証情報をリクエストした後、自分で使用することも、フェデレーティッドユーザーまたはアプリケーションに渡すこともできます。フェデレーションユーザー用に一時セキュリティ証明書をリクエストする場合、ユーザー名と、その一時セキュリティ証明書に関連付ける許可を定義した IAM ポリシーを指定する必要があります。フェデレーションユーザーに、一時証明書をリクエストした親 IAM ユーザーよりも多くの権限が付与されることはありません。

Amazon S3 にリクエストするときに、これらの一時的なセキュリティ認証情報を使用できます。API ライブラリによって、これらの認証情報を使用して必要な署名値が計算されて、リクエストが認証されます。失効した証明書を使用してリクエストを送信した場合、Amazon S3 はリクエストを拒否します。

REST API リクエストで一時セキュリティ証明書を使用してリクエストに署名する方法については、REST リクエストの署名と認証 を参照してください。AWS SDK を使用してリクエストを送信する方法については、AWS SDK を使用したリクエストの実行 を参照してください。

一時的なセキュリティ認証情報の IAM によるサポートの詳細については、IAM ユーザーガイドの「IAM の一時的なセキュリティ認証情報」を参照してください。

Amazon S3 リソースにアクセスする際のセキュリティを高めるには、バケットポリシーを設定して多要素認証 (MFA) を要求することができます。詳細については、「MFA が必要」を参照してください。Amazon S3 リソースにアクセスするために MFA を要求した後に、これらのリソースにアクセスできる唯一の方法は、MFA キーで作成された一時的な認証情報を提供することです。詳細については、AWS 多要素認証の詳細ページと IAM ユーザーガイドの MFA 保護 API アクセスの設定を参照してください。

リクエストのエンドポイント

サービスの定義済みエンドポイントに対して REST リクエストを送信します。すべての AWS のサービスとそれに対応するエンドポイントのリストについては、AWS 全般のリファレンス の「リージョンとエンドポイント」を参照してください。

IPv6 経由で Amazon S3 へのリクエストを行う

Amazon Simple Storage Service (Amazon S3) では、IPv4 プロトコルに加えて、Internet Protocol version 6 (IPv6) を使用して S3 バケットにアクセスする機能をサポートしています。Amazon S3 デュアルスタックエンドポイントは、IPv6 および IPv4 を使用した S3 バケットへのリクエストをサポートしています。IPv6 を使用して Amazon S3 にアクセスする場合、追加料金はかかりません。料金に関する詳細については、「Amazon S3 の料金」を参照してください。

トピック

- IPv6 を使用したリクエストの実行の開始方法
- IAM ポリシーでの IPv6 アドレスの使用
- IP アドレス互換性のテスト
- Amazon S3 デュアルスタックのエンドポイントの使用

IPv6 を使用したリクエストの実行の開始方法

IPv6 を使用して S3 バケットにリクエストを実行するには、デュアルスタックのエンドポイントを使用する必要があります。次のセクションでは、デュアルスタックのエンドポイントを使用した IPv6 でのリクエストの実行方法について説明します。

以下は、IPv6 でバケットにアクセスを試行する前に知っておく必要がある点です。

- バケットにアクセスするクライアントやネットワークは、IPv6 の使用を有効にする必要があります。
- 仮想ホスト形式およびパス形式のリクエストは、IPv6 アクセスをサポートしています。詳細については、「Amazon S3 デュアルスタックのエンドポイントを使用する」を参照してください。
- AWS Identity and Access Management (IAM) ユーザーまたはバケットポリシーで、ソース IP アドレスによるフィルタリングを使用する場合、IPv6 アドレス範囲を含めるようポリシーを更新する必要があります。詳細については、「IAM ポリシーでの IPv6 アドレスの使用」を参照してください。
- IPv6 を使用する場合、サーバーのアクセスログファイルは IPv6 形式の IP アドレスを出力します。IPv6 形式の `Remote IP` アドレスを解析できるように、Amazon S3 ログファイルの解析に使用する既存のツール、スクリプト、ソフトウェアを更新する必要があります。詳細については、「Amazon S3 サーバーアクセスログの形式」および「サーバーアクセスログによるリクエストのログ記録」を参照してください。

> **Note**
>
> ログファイルで IPv6 アドレスに関連する問題を見つけた場合、AWS Support にお問い合わせください。

デュアルスタックのエンドポイントを使用して IPv6 経由でリクエストを実行する

IPv6 での Amazon S3 API コールを使用したリクエストは、デュアルスタックのエンドポイントを使用して行います。Amazon S3 API オペレーションは、IPv6 経由でも IPv4 経由でも、Amazon S3 へのアクセスは同じように機能します。パフォーマンスも同じです。

REST API を使用する場合、デュアルスタックのエンドポイントに直接アクセスします。詳細については、「デュアルスタックのエンドポイント」を参照してください。

AWS Command Line Interface (AWS CLI) や AWS SDK を使用する場合、パラメータまたはフラグを使ってデュアルスタックのエンドポイントに変更できます。設定ファイルの Amazon S3 エンドポイントに上書きしてデュアルスタックのエンドポイントを直接指定することもできます。

デュアルスタックのエンドポイントを使用して、次のいずれかから IPv6 でバケットにアクセスできます。

- AWS CLI については、「AWS CLI からのデュアルスタックのエンドポイントの使用」を参照してください。
- AWS SDK については、AWS SDK からデュアルスタックのエンドポイントを使用する を参照してください。
- REST API については、REST API を使用したデュアルスタックのエンドポイントへのリクエストの実行を参照してください。

IPv6 で使用できない機能

IPv6 を介して S3 バケットにアクセスする場合、S3 バケットからの静的ウェブサイトホスティングの機能は現在サポートされていません。

IAM ポリシーでの IPv6 アドレスの使用

IPv6 を使用してバケットにアクセスを試行する前に、IP アドレスによるフィルタリングに使用される IAM ユーザーまたは S3 バケットポリシーが IPv6 アドレス範囲を含むように更新されているか確認する必要があります。IPv6 アドレスを処理するために更新されていない IP アドレスのフィルタリングポリシーでは、IPv6 の使用を開始する際、間違ってクライアントがバケットへのアクセスを取得または失う場合があります。IAM でのアクセス許可管理の詳細については、「Amazon S3 用 Identity and Access Management」を参照してください。

IP アドレスをフィルタリングする IAM ポリシーは、IP アドレス条件演算子を使用します。次のバケットポリシーは、IP アドレス条件演算子を使用して 54.240.143.* の範囲の許可された IPv4 アドレスを識別します。この範囲外のすべての IP アドレスはバケットへのアクセスを拒否されます (`examplebucket`)。すべての IPv6 アドレスは許可範囲外であるため、このポリシーは IPv6 アドレスの `examplebucket` へのアクセスをブロックします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
```

```
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

次の例のように、バケットポリシーの Condition エレメントを変更して、IPv4 (54.240.143.0/24) および IPv6 (2001:DB8:1234:5678::/64) アドレス範囲の両方を許可できます。例に示すように、IAM ユーザーとバケットポリシーの両方を更新するために同じタイプの Condition ブロックを使用できます。

```
       "Condition": {
         "IpAddress": {
            "aws:SourceIp": [
              "54.240.143.0/24",
               "2001:DB8:1234:5678::/64"
             ]
          }
        }
```

IPv6 を使用する前に、IPv6 アドレス範囲を許可する IP アドレスのフィルタリングに使用するすべての関連 IAM ユーザーとバケットポリシーを更新しなければなりません。既存の IPv4 アドレス範囲に加えて、IAM ポリシーと組織の IPv6 アドレス範囲を更新することをお勧めします。IPv6 および IPv4; でのアクセスを許可するバケットポリシーの例については、「特定の IP アドレスへのアクセスの制限」を参照してください。

IAM ユーザーポリシーは、https://console.aws.amazon.com/iam/ の IAM コンソールを使用して確認できます。IAM の詳細については、IAM ユーザーガイドを参照してください。S3 バケットポリシーの編集の詳細については、「Amazon S3 コンソールを使用したバケットポリシーの追加」を参照してください。

IP アドレス互換性のテスト

Linux/Unix または Mac OS X を使用している場合、次の例に示すように curl コマンドを使用して IPv6 でデュアルスタックのエンドポイントにアクセスできるかどうかテストできます。

Example

```
curl -v  http://s3.dualstack.us-west-2.amazonaws.com/
```

次の例のような情報を取得できます。IPv6 で接続している場合、接続されている IP アドレスは IPv6 アドレスです。

```
* About to connect() to s3-us-west-2.amazonaws.com port 80 (#0)
*   Trying IPv6 address... connected
* Connected to s3.dualstack.us-west-2.amazonaws.com (IPv6 address) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1 OpenSSL/1.0.1t
 zlib/1.2.3
> Host: s3.dualstack.us-west-2.amazonaws.com
```

Microsoft Windows 7、または Windows 10 を使用している場合、次の例に示すように ping コマンドを使用して IPv6 または IPv4 でデュアルスタックのエンドポイントにアクセスできるかどうかテストできます。

```
ping ipv6.s3.dualstack.us-west-2.amazonaws.com
```

Amazon S3 デュアルスタックのエンドポイントの使用

Amazon S3 デュアルスタックエンドポイントは、IPv6 および IPv4 を使用した S3 バケットへのリクエストをサポートしています。このセクションは、スタックのエンドポイントを使用する方法を説明します。

トピック

- Amazon S3 デュアルスタックのエンドポイントを使用する
- AWS CLI からのデュアルスタックのエンドポイントの使用
- AWS SDK からデュアルスタックのエンドポイントを使用する
- REST API からのデュアルスタックのエンドポイントの使用

Amazon S3 デュアルスタックのエンドポイントを使用する

デュアルスタックのエンドポイントにリクエストを行うと、バケット URL は IPv6 または IPv4 アドレスに解決されます。IPv6 でのバケットへのアクセスに関する詳細は、「IPv6 経由で Amazon S3 へのリクエストを行う」を参照してください。

REST API を使用する場合、エンドポイント名 (URI) を使用して直接 Amazon S3 エンドポイントにアクセスします。仮想ホスト形式またはパス形式のエンドポイント名を使用することで、デュアルスタックのエンドポイント経由で S3 バケットにアクセスできます。Amazon S3 はリージョンのデュアルスタックエンドポイント名のみをサポートしており、名前の一部としてリージョンを指定する必要があります。

デュアルスタックの仮想ホスト形式とパス形式のエンドポイント名には、次の命名規則を使用します。

- 仮想ホスティング形式のデュアルスタックのエンドポイント:

  bucketname.s3.dualstack.*aws-region*.amazonaws.com

- パス形式のデュアルスタックのエンドポイント:

  s3.dualstack.*aws-region*.amazonaws.com/*bucketname*

エンドポイント名のスタイルの詳細については、「Amazon S3 バケットに対するアクセスと一覧表示」を参照してください。Amazon S3 エンドポイントのリストは、AWS 全般のリファレンス の「リージョンとエンドポイント」を参照してください。

> **Important**
>
> デュアルスタックのエンドポイントでは、Transfer Acceleration を使用できます。詳細については、「Amazon S3 Transfer Acceleration の開始方法」を参照してください。

> **Note**
>
> Amazon S3 へのアクセスに使用する 2 つのタイプの VPC エンドポイント (インターフェイス VPC エンドポイントとゲートウェイ VPC エンドポイント) は、デュアルスタックを

サポートしていません。Amazon S3 向け VPC エンドポイントの詳細については、「AWS PrivateLink for Amazon S3 」を参照してください。

AWS Command Line Interface (AWS CLI) や AWS SDK を使用する場合、パラメータまたはフラグを使ってデュアルスタックのエンドポイントに変更できます。設定ファイルの Amazon S3 エンドポイントに上書きしてデュアルスタックのエンドポイントを直接指定することもできます。以下のセクションでは、デュアルスタックのエンドポイントを AWS CLI と AWS SDK から使用する方法を説明します。

AWS CLI からのデュアルスタックのエンドポイントの使用

このセクションでは、デュアルスタックのエンドポイントへリクエストするのに使用される AWS CLI コマンドの例を示します。AWS CLI をセットアップする手順については、「AWS CLI を使用した Amazon S3 での開発」を参照してください。

AWS Config ファイルのプロファイル内で設定値 `use_dualstack_endpoint` を `true` に設定すると、`s3` と `s3api` AWS CLI コマンドによるすべての Amazon S3 リクエストが指定されたリージョンのデュアルスタックのエンドポイントに転送されます。`--region` オプションを使用して設定ファイルまたはコマンドでリージョンを指定します。

AWS CLI でデュアルスタックのエンドポイントを使用する場合、`path` と `virtual` の両方のアドレス形式がサポートされます。バケット名がホスト名または URL の一部にある場合、設定ファイルにあるアドレス形式が統制します。デフォルトでは、CLI は可能な限り仮想形式の使用を試みますが、必要に応じてパス形式に戻します。詳細については、AWS CLI Amazon S3 の設定を参照してください。

以下の例に示すように、デフォルトプロファイルで `use_dualstack_endpoint` を `true` に、また `addressing_style` を `virtual` に設定するようコマンドを使用して構成を変更することもできます。

```
$ aws configure set default.s3.use_dualstack_endpoint true
$ aws configure set default.s3.addressing_style virtual
```

すべてのコマンドではなく、指定した AWS CLI コマンドについてのみデュアルスタックのエンドポイントを使用するには、次のいずれかの方法を使用できます。

- コマンドでデュアルスタックのエンドポイントを使用する。その場合は、すべての `--endpoint-url` または `https://s3.dualstack.aws-region.amazonaws.com` コマンドで、`http://s3.dualstack.aws-region.amazonaws.com` パラメータを `s3` または `s3api` に設定します。

```
$ aws s3api list-objects --bucket bucketname --endpoint-url https://s3.dualstack.aws-region.amazonaws.com
```

- AWS Config ファイル内で別々のプロファイルを設定する。たとえば、use_dualstack_endpoint を true に設定するプロファイルと use_dualstack_endpoint を設定しないプロファイルを作成します。コマンドを実行する際には、デュアルスタックのエンドポイントを使用するかどうかによって、適切なプロファイルを指定します。

> **Note**
>
> AWS CLI を使用していると、現在は、デュアルスタックのエンドポイントでは Transfer Acceleration を使用できません。ただし、AWS CLI は間もなくサポートされるようになります。詳細については、「AWS CLI の使用」を参照してください。

AWS SDK からデュアルスタックのエンドポイントを使用する

このセクションでは、AWS SDK を使用してデュアルスタックのエンドポイントにアクセスする方法の例を示します。

AWS SDK for Java のデュアルスタックのエンドポイントの例

次の例では、AWS SDK for Java を使用して Amazon S3 クライアントの作成時にデュアルスタックエンドポイントを有効化する方法を示します。

作業用の Java サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

public class DualStackEndpoints {

    public static void main(String[] args) {
```

```
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            // Create an Amazon S3 client with dual-stack endpoints enabled.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .withDualstackEnabled(true)
                    .build();

            s3Client.listObjects(bucketName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Windows で AWS SDK for Java を使用している場合は、必要に応じて、次の Java 仮想マシン (JVM) のプロパティを設定します。

```
java.net.preferIPv6Addresses=true
```

AWS .NET SDK のデュアルスタックのエンドポイントの例

AWS SDK for .NET を使用するときは、次の例に示すように、デュアルスタックのエンドポイントの使用を有効にするため AmazonS3Config クラスを使用します。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
```

```
    class DualStackEndpointTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            var config = new AmazonS3Config
            {
                UseDualstackEndpoint = true,
                RegionEndpoint = bucketRegion
            };
            client = new AmazonS3Client(config);
            Console.WriteLine("Listing objects stored in a bucket");
            ListingObjectsAsync().Wait();
        }

        private static async Task ListingObjectsAsync()
        {
            try
            {
                var request = new ListObjectsV2Request
                {
                    BucketName = bucketName,
                    MaxKeys = 10
                };
                ListObjectsV2Response response;
                do
                {
                    response = await client.ListObjectsV2Async(request);

                    // Process the response.
                    foreach (S3Object entry in response.S3Objects)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }
                    Console.WriteLine("Next Continuation Token: {0}",
response.NextContinuationToken);
                    request.ContinuationToken = response.NextContinuationToken;
                } while (response.IsTruncated == true);
            }
```

```
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

オブジェクトのリストの .NET のサンプルの一覧については、プログラムによるオブジェクトキーのリスト化を参照してください。

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

REST API からのデュアルスタックのエンドポイントの使用

REST API を使用したデュアルスタックのエンドポイントへのリクエストについては、「REST API を使用したデュアルスタックのエンドポイントへのリクエストの実行」を参照してください。

AWS SDK を使用したリクエストの実行

トピック

認証リクエストを Amazon S3 に送信するには、AWS SDK を使用するか、アプリケーションで直接 REST API を呼び出します。AWS SDK API は、指定した証明書を使用して認証用の署名を計算します。アプリケーションで直接 REST API を使用する場合、リクエストを認証するために、署名の計算に必要なコードを記述する必要があります。利用可能な AWS SDK のリストについては、サンプルコードとライブラリを参照してください。

AWS アカウント または IAM ユーザーの認証情報を使用したリクエストの実行

AWS アカウント または IAM ユーザーのセキュリティ認証情報を使用して、認証リクエストを Amazon S3 に送信できます。このセクションでは、AWS SDK for Java、AWS SDK for .NET、および AWS SDK for PHP を使用して認証リクエストを送信する方法の例を示します。利用可能な AWS SDK のリストについては、サンプルコードとライブラリを参照してください。

各 AWS SDK は、SDK 固有の認証情報プロバイダチェーンを使用して認証情報の検索を行い、その認証情報を使用して認証情報の所有者に代わってアクションを実行します。これらの認証情報プロバイダーチェーンに共通していることは、ユーザーのローカルの AWS 認証情報ファイルを探すことです。

詳細については、以下のトピックをご参照ください。

トピック

ローカルの AWS 認証情報ファイルを作成するには

AWS SDK 用の認証情報を設定する最も簡単な方法は、AWS 認証情報ファイルを使用することです。AWS Command Line Interface (AWS CLI) を使用している場合、既にローカルの AWS 認証情報

ファイルを設定済みかもしれません。そうでない場合は、以下の手順で認証情報ファイルを設定します。

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. コードがアクセス許可を使用する必要があるサービスやアクションに限定した許可を持つ、新しいユーザーを作成します。新しいユーザーの作成方法の詳細については、「IAM ユーザーの作成 (コンソール)」を参照し、ステップ 8 までの手順に従ってください。
3. AWS 認証情報のローカルコピーを保存するには、[Download .csv] (.csv のダウンロード) を選択します。
4. コンピュータでホームディレクトリに移動し `.aws` ディレクトリを作成します。Linux または OS X のような Unix ベースのシステムでは、次の場所になります。

   ```
   ~/.aws
   ```

   Windows では、次の場所になります。

   ```
   %HOMEPATH%\.aws
   ```

5. `.aws` ディレクトリで `credentials` という名前の新しいファイルを作成します。
6. IAM コンソールからダウンロードした認証情報の `.csv` ファイルを開き、その内容を以下の形式で `credentials` ファイル内にコピーします。

   ```
   [default]
   aws_access_key_id = your_access_key_id
   aws_secret_access_key = your_secret_access_key
   ```

7. `credentials` ファイルを保存し、ステップ 3 でダウンロードした `.csv` ファイルを削除します。

これで共有認証情報ファイルがローカルコンピュータで設定され、AWS SDK で使用する準備ができました。

AWS SDK を使用した、認証されたリクエストの送信

AWS SDK を使用して認証されたリクエストを送信します。認証済みリクエストの送信の詳細については、「AWS セキュリティ認証情報」または「IAM Identity Center 認証」を参照してください。

Java

AWS アカウント または IAM ユーザーの認証情報を使用して、認証済みのリクエストを Amazon S3 に送信するには、以下を実行します。

- `AmazonS3ClientBuilder` クラスを使用して `AmazonS3Client` インスタンスを作成します。
- `AmazonS3Client` メソッドのいずれかを実行して、Amazon S3 にリクエストを送信します。クライアントにより、指定した認証情報から必要な署名が生成されて、リクエストに追加されます。

次の例では、前のタスクを実行します。作業サンプルの作成およびテストについては、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

Example

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsRequest;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.s3.model.S3ObjectSummary;

import java.io.IOException;
import java.util.List;

public class MakingRequests {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();
```

```
            // Get a list of objects in the bucket, two at a time, and
            // print the name and size of each object.
            ListObjectsRequest listRequest = new
ListObjectsRequest().withBucketName(bucketName).withMaxKeys(2);
            ObjectListing objects = s3Client.listObjects(listRequest);
            while (true) {
                List<S3ObjectSummary> summaries = objects.getObjectSummaries();
                for (S3ObjectSummary summary : summaries) {
                    System.out.printf("Object \"%s\" retrieved with size %d\n",
summary.getKey(), summary.getSize());
                }
                if (objects.isTruncated()) {
                    objects = s3Client.listNextBatchOfObjects(objects);
                } else {
                    break;
                }
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

AWS アカウント または IAM ユーザーの認証情報を使用して、認証済みのリクエストを送信するには

- AmazonS3Client クラスのインスタンスを作成します。
- AmazonS3Client メソッドのいずれかを実行して、Amazon S3 にリクエストを送信します。クライアントにより、指定した認証情報から必要な署名が生成されて、Amazon S3 に送信されるリクエストに追加されます。

詳細については、「AWS アカウント または IAM ユーザーの認証情報を使用したリクエストの実行」を参照してください。

> **Note**
>
> - AmazonS3Client クライアントは、セキュリティ認証情報を指定せずに作成できます。このクライアントを使用して送信されるリクエストは、署名なしの匿名リクエストです。公開されていないリソースに対して匿名リクエストを送信すると、Amazon S3 はエラーを返します。
> - AWS アカウントを作成し、必要なユーザーを作成できます。また、これらのユーザーの認証情報も管理できます。以下の例のタスクを実行するときは、これらの認証情報が必要になります。詳細については、AWS SDK for .NET デベロッパーガイドの「AWS 認証情報の設定」を参照してください。
>
> その後、プロファイルと認証情報をアクティブに取得して、AWS のサービスのクライアントの作成時にそれら認証情報を明示的に使用するように、アプリケーションを設定することもできます。詳細については、AWS SDK for .NET デベロッパーガイドのアプリケーションの認証情報およびプロファイルへのアクセスを参照してください。

次の C# の例では、前のタスクを実行する方法を示します。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class MakeS3RequestTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
```

```
        {
            using (client = new AmazonS3Client(bucketRegion))
            {
                Console.WriteLine("Listing objects stored in a bucket");
                ListingObjectsAsync().Wait();
            }
        }

        static async Task ListingObjectsAsync()
        {
            try
            {
                ListObjectsRequest request = new ListObjectsRequest
                {
                    BucketName = bucketName,
                    MaxKeys = 2
                };
                do
                {
                    ListObjectsResponse response = await
client.ListObjectsAsync(request);
                    // Process the response.
                    foreach (S3Object entry in response.S3Objects)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }

                    // If the response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.Marker = response.NextMarker;
                    }
                    else
                    {
                        request = null;
                    }
                } while (request != null);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
```

```
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

使用例については、「Amazon S3 オブジェクトの概要」および「バケットの概要」を参照してください。これらの例をテストするには、AWS アカウント または IAM ユーザーの認証情報を使用します。

たとえば、バケットのオブジェクトキーをすべて表示する方法については、プログラムによるオブジェクトキーのリスト化 を参照してください。

PHP

このセクションでは、バージョン 3 の AWS SDK for PHP のクラスで AWS アカウント または IAM ユーザー認証情報を使用して、認証済みのリクエストを送信する方法について説明します。AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

次の PHP の例では、クライアントでセキュリティ認証情報を使用して、アカウントのバケットを一覧表示するリクエストを実行する方法を示します。

Example

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
]);
```

```
// Retrieve the list of buckets.
$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // Print the list of objects to the page.
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

> **Note**
>
> S3Client クライアントは、セキュリティ認証情報を指定せずに作成できます。このクライアントを使用して送信されるリクエストは、署名なしの匿名リクエストです。公開されていないリソースに対して匿名リクエストを送信すると、Amazon S3 はエラーを返します。詳細については、AWS SDK for PHP ドキュメントの匿名クライアントの作成を参照してください。

実例については、Amazon S3 オブジェクトの概要 を参照してください。これらの例をテストするには、AWS アカウント または IAM ユーザーの認証情報を使用します。

バケット内のオブジェクトキーをリストする例については、「プログラムによるオブジェクトキーのリスト化」を参照してください。

Ruby

バージョン 3 の AWS SDK for Ruby を使用して Amazon S3 を呼び出す前に、SDK がバケットやオブジェクトへのアクセスの確認に使用する AWS アクセス認証情報を設定する必要があります。ローカルシステムの AWS 認証情報プロファイルで設定した認証情報を共有している場合、バージョン 3 の SDK for Ruby では、コード内で宣言することなく、これらの認証情報を使用で

きます。共有認証情報の設定の詳細については、AWS アカウント または IAM ユーザーの認証情報を使用したリクエストの実行 を参照してください。

次の Ruby コードスニペットは、ローカルコンピュータ上の共有 AWS 認証情報ファイルの認証情報を使用して、特定のバケット内のすべてのオブジェクトキー名を取得するリクエストを認証します。以下が実行されます。

1. `Aws::S3::Client` クラスのインスタンスを作成します。
2. `list_objects_v2` の `Aws::S3::Client` メソッドを使用してバケット内のオブジェクトを列挙することで、Amazon S3 へのリクエストを実行します。クライアントは、コンピュータ上の AWS 認証情報ファイルの認証情報から必要な署名値を生成し、それを Amazon S3 に送信するリクエストに含めます。
3. オブジェクトキー名の配列をターミナルに出力します。

Example

```
# Prerequisites:
#   - An existing Amazon S3 bucket.

require "aws-sdk-s3"

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'doc-example-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts "The object keys in this bucket are (first 50 objects):"
    objects.contents.each do |object|
      puts object.key
    end
  else
```

```
    puts "No objects found in this bucket."
  end

  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  return false
end

# Example usage:
def run_me
  region = "us-west-2"
  bucket_name = "BUCKET_NAME"
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

ローカルの AWS 認証情報ファイルがない場合でも、Aws::S3::Client リソースを作成し、Amazon S3 のバケットとオブジェクトに対してコードを実行できます。バージョン 3 の SDK for Ruby を使用して送信されるリクエストは匿名であり、デフォルトでは署名はありません。公開されていないリソースに対して匿名リクエストを送信すると、Amazon S3 はエラーを返します。

次のより堅牢な例のように、SDK for Ruby アプリケーション用の前述のコードスニペットを使用して展開することができます。

```
# Prerequisites:
#  - An existing Amazon S3 bucket.

require "aws-sdk-s3"

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'doc-example-bucket')
def list_bucket_objects?(s3_client, bucket_name)
```

```
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts "The object keys in this bucket are (first 50 objects):"
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts "No objects found in this bucket."
  end

  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  return false
end

# Example usage:
def run_me
  region = "us-west-2"
  bucket_name = "BUCKET_NAME"
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

Go

Example

次の例では、SDK for Go が共有認証情報ファイルから自動的に読み込む AWS 認証情報を使用しています。

```
package main

import (
```

```
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 s3Client := s3.NewFromConfig(sdkConfig)
 count := 10
 fmt.Printf("Let's list up to %v buckets for your account.\n", count)
 result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
 if err != nil {
  fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Buckets) == 0 {
  fmt.Println("You don't have any buckets!")
 } else {
  if count > len(result.Buckets) {
   count = len(result.Buckets)
  }
  for _, bucket := range result.Buckets[:count] {
   fmt.Printf("\t%v\n", *bucket.Name)
  }
 }
}
```

関連リソース

- AWS SDK を使用した Amazon S3 での開発

- Amazon S3 向け AWS SDK for PHP Aws\S3\S3Client クラスの場合
- AWS SDK for PHP ドキュメント

IAM ユーザーの一時的な認証情報を使用したリクエストの実行

AWS アカウント または IAM ユーザーは、一時的なセキュリティ認証情報をリクエストし、そのセキュリティ認証情報を使用して、認証リクエストを Amazon S3 に送信できます。このセクションでは、AWS SDK for Java、.NET、および PHP を使用して一時セキュリティ認証情報を取得する方法と、Amazon S3 に対するリクエストの認証にその認証情報を使用する方法について説明します。

Java

IAM ユーザーまたは AWS アカウント は、AWS SDK for Java を使用して一時的なセキュリティ認証情報をリクエストし (リクエストの実行 を参照)、その認証情報を使用して Amazon S3 にアクセスできます。これらの認証情報は、指定したセッションの有効期間が過ぎると失効します。

デフォルトでは、セッションの有効期間は 1 時間です。IAM ユーザーの認証情報を使用する場合、一時的なセキュリティ認証情報をリクエストする期間を 15 分からロールの最大セッション期間までで指定できます。一時的なセキュリティ認証情報の詳細については、IAM ユーザーガイドの「IAM の一時的なセキュリティ認証情報」を参照してください。リクエストの実行の詳細については、「リクエストの実行」を参照してください。

一時的なセキュリティ認証情報を取得して Amazon S3 にアクセスするには

1. `AWSSecurityTokenService` クラスのインスタンスを作成します。認証情報の指定の詳細については、「AWS SDK を使用した Amazon S3 での開発」をご覧ください。
2. Security Token Service (STS) クライアントの `assumeRole()` メソッドを呼び出して、該当ロールの一時的なセキュリティ認証情報を取得します。
3. 一時的なセキュリティ認証情報を `BasicSessionCredentials` オブジェクトにパッケージ化します。このオブジェクトを使用して、一時的なセキュリティ認証情報を Amazon S3 クライアントに提供します。
4. 一時的なセキュリティ認証情報を使用して、`AmazonS3Client` クラスのインスタンスを作成します。このクライアントを使用してリクエストを Amazon S3 に送信します。失効した認証情報を使用してリクエストを送信すると、Amazon S3 はエラーを返します。

> **Note**
>
> AWS アカウント のセキュリティ認証情報を使用して一時的なセキュリティ認証情報を取得した場合、この認証情報は 1 時間だけ有効です。セッションの有効期間を指定できるのは、セッションのリクエストに IAM ユーザー証明書を使用する場合のみです。

次の例では、指定したバケット内のオブジェクトキーを一覧表示します。この例では、セッションの一時的なセキュリティ認証情報を取得し、これを使用して認証済みのリクエストを Amazon S3 に送信します。

IAM ユーザーの認証情報を使用してサンプルをテストする場合は、AWS アカウントで IAM ユーザーを作成する必要があります。IAM ユーザーを作成する方法の詳細については、IAM ユーザーガイドの「最初の IAM ユーザーおよび管理者グループの作成」を参照してください。

作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.AssumeRoleRequest;
import com.amazonaws.services.securitytoken.model.AssumeRoleResult;
import com.amazonaws.services.securitytoken.model.Credentials;

public class MakingRequestsWithIAMTempCredentials {
    public static void main(String[] args) {
        String clientRegion = "*** Client region ***";
        String roleARN = "*** ARN for role to be assumed ***";
        String roleSessionName = "*** Role session name ***";
        String bucketName = "*** Bucket name ***";

        try {
            // Creating the STS client is part of your trusted code. It has
            // the security credentials you use to obtain temporary security
 credentials.
            AWSSecurityTokenService stsClient =
 AWSSecurityTokenServiceClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();
```

```
            // Obtain credentials for the IAM role. Note that you cannot assume the
role of
            // an AWS root account;
            // Amazon S3 will deny access. You must use credentials for an IAM user
or an
            // IAM role.
            AssumeRoleRequest roleRequest = new AssumeRoleRequest()
                    .withRoleArn(roleARN)
                    .withRoleSessionName(roleSessionName);
            AssumeRoleResult roleResponse = stsClient.assumeRole(roleRequest);
            Credentials sessionCredentials = roleResponse.getCredentials();

            // Create a BasicSessionCredentials object that contains the credentials
you
            // just retrieved.
            BasicSessionCredentials awsCredentials = new BasicSessionCredentials(
                    sessionCredentials.getAccessKeyId(),
                    sessionCredentials.getSecretAccessKey(),
                    sessionCredentials.getSessionToken());

            // Provide temporary security credentials so that the Amazon S3 client
            // can send authenticated requests to Amazon S3. You create the client
            // using the sessionCredentials object.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new
AWSStaticCredentialsProvider(awsCredentials))
                    .withRegion(clientRegion)
                    .build();

            // Verify that assuming the role worked and the permissions are set
correctly
            // by getting a set of object keys from the bucket.
            ObjectListing objects = s3Client.listObjects(bucketName);
            System.out.println("No. of Objects: " +
objects.getObjectSummaries().size());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
```

```
    }
}
```

.NET

IAM ユーザーまたは AWS アカウント は、AWS SDK for .NET を使用して一時的なセキュリティ認証情報をリクエストし、その認証情報を使用して Amazon S3 にアクセスできます。これらの認証情報は、セッションの有効期間が過ぎると失効します。

デフォルトでは、セッションの有効期間は 1 時間です。IAM ユーザーの認証情報を使用する場合、一時的なセキュリティ認証情報をリクエストする期間を 15 分からロールの最大セッション期間までで指定できます。一時的なセキュリティ認証情報の詳細については、IAM ユーザーガイドの「IAM の一時的なセキュリティ認証情報」を参照してください。リクエストの実行の詳細については、「リクエストの実行」を参照してください。

一時的なセキュリティ認証情報を取得して Amazon S3 にアクセスするには

1. AWS Security Token Service クライアント `AmazonSecurityTokenServiceClient` のインスタンスを作成します。認証情報の指定の詳細については、「AWS SDK を使用した Amazon S3 での開発」をご覧ください。
2. 前の手順で作成した STS クライアントの `GetSessionToken` メソッドを呼び出して、セッションを開始します。`GetSessionTokenRequest` オブジェクトを使用して、このメソッドにセッション情報を指定します。

   このメソッドは一時的なセキュリティ認証情報を返します。
3. `SessionAWSCredentials` オブジェクトのインスタンスに一時セキュリティ証明書をパッケージ化します。このオブジェクトを使用して、一時的なセキュリティ認証情報を Amazon S3 クライアントに提供します。
4. 一時セキュリティ証明書を渡して、`AmazonS3Client` クラスのインスタンスを作成します。このクライアントを使用してリクエストを Amazon S3 に送信します。失効した認証情報を使用してリクエストを送信すると、Amazon S3 はエラーを返します。

> **Note**
>
> AWS アカウント のセキュリティ認証情報を使用して一時的なセキュリティ認証情報を取得した場合、この認証情報は 1 時間だけ有効です。セッションの有効期間を指定でき

るのは、IAM ユーザーの認証情報を使用してセッションをリクエストする場合に限ります。

次の C# の例では、指定したバケットのオブジェクトキーを一覧表示します。この例では、わかりやすいように、デフォルトの 1 時間のセッションの一時セキュリティ認証情報を取得し、それを使用して認証リクエストを Amazon S3 に送信しています。

IAM ユーザーの認証情報を使用してサンプルをテストする場合は、AWS アカウントで IAM ユーザーを作成する必要があります。IAM ユーザーを作成する方法の詳細については、IAM ユーザーガイドの「最初の IAM ユーザーおよび管理者グループの作成」を参照してください。リクエストの実行の詳細については、「リクエストの実行」を参照してください。

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempCredExplicitSessionStartTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
```

```
            try
            {
                // Credentials use the default AWS SDK for .NET credential search
chain.
                // On local development machines, this is your default profile.
                Console.WriteLine("Listing objects stored in a bucket");
                SessionAWSCredentials tempCredentials = await
GetTemporaryCredentialsAsync();

                // Create a client by providing temporary security credentials.
                using (s3Client = new AmazonS3Client(tempCredentials, bucketRegion))
                {
                    var listObjectRequest = new ListObjectsRequest
                    {
                        BucketName = bucketName
                    };
                    // Send request to Amazon S3.
                    ListObjectsResponse response = await
s3Client.ListObjectsAsync(listObjectRequest);
                    List<S3Object> objects = response.S3Objects;
                    Console.WriteLine("Object count = {0}", objects.Count);
                }
            }
            catch (AmazonS3Exception s3Exception)
            {
                Console.WriteLine(s3Exception.Message, s3Exception.InnerException);
            }
            catch (AmazonSecurityTokenServiceException stsException)
            {
                Console.WriteLine(stsException.Message,
stsException.InnerException);
            }
        }

        private static async Task<SessionAWSCredentials>
GetTemporaryCredentialsAsync()
        {
            using (var stsClient = new AmazonSecurityTokenServiceClient())
            {
                var getSessionTokenRequest = new GetSessionTokenRequest
                {
                    DurationSeconds = 7200 // seconds
                };
```

```
                GetSessionTokenResponse sessionTokenResponse =
                            await
 stsClient.GetSessionTokenAsync(getSessionTokenRequest);

                Credentials credentials = sessionTokenResponse.Credentials;

                var sessionCredentials =
                    new SessionAWSCredentials(credentials.AccessKeyId,
                                              credentials.SecretAccessKey,
                                              credentials.SessionToken);
                return sessionCredentials;
            }
        }
    }
}
```

PHP

AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

IAM ユーザーまたは AWS アカウント は、バージョン 3 の AWS SDK for PHP を使用して一時的なセキュリティ認証情報をリクエストできます。次に、この一時的な認証情報を使用して Amazon S3 にアクセスできます。認証情報は、セッションの有効期間が失効すると同時に失効します。

デフォルトでは、セッションの有効期間は 1 時間です。IAM ユーザーの認証情報を使用する場合、一時的なセキュリティ認証情報をリクエストする期間を 15 分からロールの最大セッション期間までで指定できます。一時的なセキュリティ認証情報の詳細については、IAM ユーザーガイドの「IAM の一時的なセキュリティ認証情報」を参照してください。リクエストの実行の詳細については、「リクエストの実行」を参照してください。

> **Note**
>
> AWS アカウント のセキュリティ認証情報を使用して一時的なセキュリティ認証情報を取得する場合、取得した一時的なセキュリティ認証情報は 1 時間だけ有効です。セッションの有効期間を指定できるのは、セッションのリクエストに IAM ユーザー証明書を使用する場合のみです。

Example

次の PHP の例では、一時的なセキュリティ認証情報を使用して、指定したバケットのオブジェクトキーを一覧表示します。この例では、一時的なセキュリティ認証情報をデフォルトの 1 時間のセッションとして取得し、それを使用して認証済みのリクエストを Amazon S3 に送信します。AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

IAM ユーザーの認証情報を使用して例をテストする場合は、AWS アカウントで IAM ユーザーを作成する必要があります。IAM ユーザーを作成する方法の詳細については、IAM ユーザーガイドの「最初の IAM ユーザーおよび管理者グループの作成」を参照してください。IAM ユーザーの認証情報を使用してセッションをリクエストするときにセッションの有効期間を設定する例については、「IAM ユーザーの一時的な認証情報を使用したリクエストの実行 」を参照してください。

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

$sessionToken = $sts->getSessionToken();

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key'    => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token'  => $sessionToken['Credentials']['SessionToken']
    ]
]);

$result = $s3->listBuckets();

try {
```

```
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // List objects
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

IAM ユーザーまたは AWS アカウント は、AWS SDK for Ruby を使用して一時的なセキュリティ認証情報をリクエストし、その認証情報を使用して Amazon S3 にアクセスできます。これらの認証情報は、セッションの有効期間が過ぎると失効します。

デフォルトでは、セッションの有効期間は 1 時間です。IAM ユーザーの認証情報を使用する場合、一時的なセキュリティ認証情報をリクエストする期間を 15 分からロールの最大セッション期間までで指定できます。一時的なセキュリティ認証情報の詳細については、IAM ユーザーガイドの「IAM の一時的なセキュリティ認証情報」を参照してください。リクエストの実行の詳細については、「リクエストの実行」を参照してください。

> **Note**
>
> AWS アカウント のセキュリティ認証情報を使用して一時的なセキュリティ認証情報を取得する場合、取得した一時的なセキュリティ認証情報は 1 時間だけ有効です。セッションの有効期間を指定できるのは、セッションのリクエストに IAM ユーザー認証情報を使用する場合のみです。

次の Ruby の例では、有効期間が 1 時間の一時的なユーザーを作成し、指定したバケットの項目を一覧表示します。この例を使用するには、新しい AWS Security Token Service (AWS STS) クライアントを作成し、Amazon S3 バケットを一覧表示するための許可が AWS 認証情報に必要です。

```
# Prerequisites:
# - A user in AWS Identity and Access Management (IAM). This user must
#   be able to assume the following IAM role. You must run this code example
#   within the context of this user.
# - An existing role in IAM that allows all of the Amazon S3 actions for all of the
#   resources in this code example. This role must also trust the preceding IAM
 user.
# - An existing S3 bucket.

require "aws-sdk-core"
require "aws-sdk-s3"
require "aws-sdk-iam"

# Checks whether a user exists in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Boolean] true if the user exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless user_exists?(iam_client, 'my-user')
def user_exists?(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return true if response.user.user_name
rescue Aws::IAM::Errors::NoSuchEntity
  # User doesn't exist.
rescue StandardError => e
  puts "Error while determining whether the user " \
    "'#{user_name}' exists: #{e.message}"
end

# Creates a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS:IAM::Types::User] The new user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = create_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def create_user(iam_client, user_name)
  response = iam_client.create_user(user_name: user_name)
```

```
  return response.user
rescue StandardError => e
  puts "Error while creating the user '#{user_name}': #{e.message}"
end

# Gets a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS:IAM::Types::User] The existing user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def get_user(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return response.user
rescue StandardError => e
  puts "Error while getting the user '#{user_name}': #{e.message}"
end

# Checks whether a role exists in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The role's name.
# @return [Boolean] true if the role exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless role_exists?(iam_client, 'my-role')
def role_exists?(iam_client, role_name)
  response = iam_client.get_role(role_name: role_name)
  return true if response.role.role_name
rescue StandardError => e
  puts "Error while determining whether the role " \
    "'#{role_name}' exists: #{e.message}"
end

# Gets credentials for a role in IAM.
#
# @param sts_client [Aws::STS::Client] An initialized AWS STS client.
# @param role_arn [String] The role's Amazon Resource Name (ARN).
# @param role_session_name [String] A name for this role's session.
# @param duration_seconds [Integer] The number of seconds this session is valid.
# @return [AWS::AssumeRoleCredentials] The credentials.
```

```
# @example
#   sts_client = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_credentials(
#     sts_client,
#     'arn:aws:iam::123456789012:role/AmazonS3ReadOnly',
#     'ReadAmazonS3Bucket',
#     3600
#   )
#   exit 1 if credentials.nil?
def get_credentials(sts_client, role_arn, role_session_name, duration_seconds)
  Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: role_session_name,
    duration_seconds: duration_seconds
  )
rescue StandardError => e
  puts "Error while getting credentials: #{e.message}"
end

# Checks whether a bucket exists in Amazon S3.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The name of the bucket.
# @return [Boolean] true if the bucket exists; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless bucket_exists?(s3_client, 'doc-example-bucket')
def bucket_exists?(s3_client, bucket_name)
  response = s3_client.list_buckets
  response.buckets.each do |bucket|
    return true if bucket.name == bucket_name
  end
rescue StandardError => e
  puts "Error while checking whether the bucket '#{bucket_name}' " \
    "exists: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
```

```
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'doc-example-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts "Name => ETag"
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end
```

関連リソース

- AWS SDK を使用した Amazon S3 での開発
- Amazon S3 向け AWS SDK for PHP Aws\S3\S3Client クラスの場合
- AWS SDK for PHP ドキュメント

フェデレーションユーザーの一時的な認証情報を使用したリクエストの実行

一時的なセキュリティ認証情報をリクエストし、AWS リソースにアクセスする必要があるフェデレーションユーザーまたはアプリケーションにその認証情報を提供できます。このセクションでは、AWS SDK を使用して、フェデレーションユーザーまたはアプリケーションのために一時セキュリティ証明書を取得し、その証明書を使用して認証リクエストを Amazon S3 に送信する方法の例について説明します。利用可能な AWS SDK のリストについては、サンプルコードとライブラリを参照してください。

> **Note**
>
> AWS アカウント と IAM ユーザーのいずれも、フェデレーティッドユーザーの一時的なセキュリティ認証情報をリクエストできます。ただし、より高度なセキュリティのためには、必要な権限を持つ IAM ユーザーのみが、この一時証明書をリクエストできるようにすることをお勧めします。こうすることで、フェデレーションユーザーに付与される権限が、リクエストを送信した IAM ユーザーの権限以下となります。アプリケーションによっては、フェデレーティッドユーザーおよびアプリケーションに一時的なセキュリティ認証情報を付与することを唯一の目的として、特定のアクセス許可を持つ IAM ユーザーを作成する方が適切な場合もあります。

Java

フェデレーティッドユーザーおよびアプリケーションから AWS リソースへのアクセスのリクエストが認証されて送信されるように、フェデレーティッドユーザーおよびアプリケーションに一時的なセキュリティ認証情報を提供することができます。このような一時的な認証情報をリクエストする場合、ユーザー名と、付与するリソースに対するアクセス許可を示す IAM ポリシーを指定する必要があります。デフォルトでは、セッションの有効期間は 1 時間です。フェデレーションユーザーおよびアプリケーション用に一時的なセキュリティ認証情報をリクエストするときに、明示的に別の有効期間値を設定できます。

> **Note**
>
> フェデレーティッドユーザーおよびアプリケーション用の一時的なセキュリティ認証情報をリクエストする際のセキュリティ強化策として、必要なアクセス許可のみを持つ専用の IAM ユーザーを使用することをお勧めします。作成した一時ユーザーに、一時的なセキュリティ認証情報をリクエストした IAM ユーザーより多くの権限が付与されることは

ありません。詳細については、「AWS Identity and Access Management に関するよくある質問」を参照してください。

セキュリティ認証情報を提供し、認証されたリクエストを送信してリソースにアクセスするには、以下の操作を行います。

- AWSSecurityTokenServiceClient クラスのインスタンスを作成します。
- Security Token Service (STS) クライアントの getFederationToken( ) メソッドを呼び出してセッションを開始します。一時的な認証情報にアタッチするユーザー名と IAM ポリシーなどのセッション情報を指定します。必要に応じて、セッションの有効期間を指定できます。このメソッドは一時的なセキュリティ認証情報を返します。
- BasicSessionCredentials オブジェクトのインスタンスに一時セキュリティ証明書をパッケージ化します。このオブジェクトを使用して、一時的なセキュリティ認証情報を Amazon S3 クライアントに提供します。
- 一時的なセキュリティ認証情報を使用して、AmazonS3Client クラスのインスタンスを作成します。このクライアントを使用してリクエストを Amazon S3 に送信します。失効した認証情報を使用してリクエストを送信すると、Amazon S3 はエラーを返します。

Example

例では、指定された S3 バケットのキーをリストしています。この例では、フェデレーティッドユーザー用に 2 時間のセッションの一時的なセキュリティ認証情報を取得し、それを使用して認証リクエストを Amazon S3 に送信します。例を実行するには、ユーザーに一時的なセキュリティ認証情報のリクエストと AWS リソースの一覧表示を許可するアタッチされたポリシーで、IAM ユーザーを作成する必要があります。次のポリシーでは、これが達成されます。

```
{
  "Statement":[{
      "Action":["s3:ListBucket",
        "sts:GetFederationToken*"
      ],
      "Effect":"Allow",
      "Resource":"*"
    }
  ]
}
```

IAM ユーザーを作成する方法の詳細については、IAM ユーザーガイドの「最初の IAM ユーザーおよび管理者グループの作成」を参照してください。

IAM ユーザーを作成し、前述のポリシーをアタッチしたら、以下の例を実行できます。作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.S3Actions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;

import java.io.IOException;

public class MakingRequestsWithFederatedTempCredentials {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Specify bucket name ***";
        String federatedUser = "*** Federated user name ***";
        String resourceARN = "arn:aws:s3:::" + bucketName;

        try {
            AWSSecurityTokenService stsClient = AWSSecurityTokenServiceClientBuilder
                    .standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
```

```
                    .build();

            GetFederationTokenRequest getFederationTokenRequest = new
GetFederationTokenRequest();
            getFederationTokenRequest.setDurationSeconds(7200);
            getFederationTokenRequest.setName(federatedUser);

            // Define the policy and add it to the request.
            Policy policy = new Policy();
            policy.withStatements(new Statement(Effect.Allow)
                    .withActions(S3Actions.ListObjects)
                    .withResources(new Resource(resourceARN)));
            getFederationTokenRequest.setPolicy(policy.toJson());

            // Get the temporary security credentials.
            GetFederationTokenResult federationTokenResult =
stsClient.getFederationToken(getFederationTokenRequest);
            Credentials sessionCredentials = federationTokenResult.getCredentials();

            // Package the session credentials as a BasicSessionCredentials
            // object for an Amazon S3 client object to use.
            BasicSessionCredentials basicSessionCredentials = new
BasicSessionCredentials(
                    sessionCredentials.getAccessKeyId(),
                    sessionCredentials.getSecretAccessKey(),
                    sessionCredentials.getSessionToken());
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new
AWSStaticCredentialsProvider(basicSessionCredentials))
                    .withRegion(clientRegion)
                    .build();

            // To verify that the client works, send a listObjects request using
            // the temporary security credentials.
            ObjectListing objects = s3Client.listObjects(bucketName);
            System.out.println("No. of Objects = " +
objects.getObjectSummaries().size());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }
}
```

.NET

フェデレーティッドユーザーおよびアプリケーションから AWS リソースへのアクセスのリクエストが認証されて送信されるように、フェデレーティッドユーザーおよびアプリケーションに一時的なセキュリティ認証情報を提供することができます。このような一時的な認証情報をリクエストする場合、ユーザー名と、付与するリソースに対するアクセス許可を示す IAM ポリシーを指定する必要があります。デフォルトでは、セッションの有効期間は 1 時間です。フェデレーションユーザーおよびアプリケーション用に一時的なセキュリティ認証情報をリクエストするときに、明示的に別の有効期間値を設定できます。認証されたリクエストを送信する方法については、「リクエストの実行」を参照してください。

> **Note**
>
> フェデレーティッドユーザーおよびアプリケーション用の一時的なセキュリティ認証情報をリクエストする際のセキュリティ強化策として、必要なアクセス許可のみを持つ専用の IAM ユーザーを使用することをお勧めします。作成した一時ユーザーに、一時的なセキュリティ認証情報をリクエストした IAM ユーザーより多くの権限が付与されることはありません。詳細については、「AWS Identity and Access Management に関するよくある質問」を参照してください。

次の作業を行います。

- AWS Security Token Service クライアント `AmazonSecurityTokenServiceClient` クラスのインスタンスを作成します。
- STS クライアントの `GetFederationToken` メソッドを呼び出してセッションを開始します。一時的な認証情報にアタッチするユーザー名と IAM ポリシーなどのセッション情報を指定する必要があります。必要に応じて、セッションの有効期間を指定できます。このメソッドは一時的なセキュリティ認証情報を返します。
- `SessionAWSCredentials` オブジェクトのインスタンスに一時セキュリティ証明書をパッケージ化します。このオブジェクトを使用して、一時的なセキュリティ認証情報を Amazon S3 クライアントに提供します。

- 一時的なセキュリティ認証情報を渡して、AmazonS3Client クラスのインスタンスを作成します。このクライアントを使用してリクエストを Amazon S3 に送信します。失効した認証情報を使用してリクエストを送信すると、Amazon S3 はエラーを返します。

Example

次の C# の例では、指定したバケットのキーをリストします。この例では、フェデレーティッドユーザー (User1) 用に 2 時間のセッションの一時的なセキュリティ認証情報を取得し、それを使用して認証リクエストを Amazon S3 に送信します。

- この演習では、最小限のアクセス許可を持つ IAM ユーザーを作成します。この IAM ユーザーの認証情報を使用して、他のユーザーの一時的な認証情報をリクエストします。この例では、特定のバケットに含まれるオブジェクトのみ表示されています。以下のポリシーをアタッチして IAM ユーザーを作成します。

```
{
  "Statement":[{
      "Action":["s3:ListBucket",
        "sts:GetFederationToken*"
      ],
      "Effect":"Allow",
      "Resource":"*"
    }
  ]
}
```

このポリシーでは、IAM ユーザーが一時的なセキュリティ認証情報と、AWS リソースを一覧表示できるだけの許可をリクエストすることを許可しています。IAM ユーザーを作成する方法の詳細については、IAM ユーザーガイドの「最初の IAM ユーザーおよび管理者グループの作成」を参照してください。

- IAM ユーザーのセキュリティ認証情報を使用して、以下の例をテストします。この例では、一時的なセキュリティ認証情報を使用して、認証リクエストを Amazon S3 に送信します。この例では、フェデレーティッドユーザー (User1) 用に、アクセスを特定のバケット (YourBucketName) 内のオブジェクトの列挙に制限する一時的なセキュリティ認証情報をリクエストするときに、以下のポリシーを指定します。ポリシーを更新し、独自の既存のバケット名を指定する必要があります。

```
{
  "Statement":[
```

```
    {
      "Sid":"1",
      "Action":["s3:ListBucket"],
      "Effect":"Allow",
      "Resource":"arn:aws:s3:::YourBucketName"
    }
  ]
}
```

- Example

  以下の例を編集して、前述のフェデレーティッドユーザーアクセスポリシーで指定したバケット名を指定します。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempFederatedCredentialsTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
            try
```

```
            {
                Console.WriteLine("Listing objects stored in a bucket");
                // Credentials use the default AWS SDK for .NET credential search
chain.
                // On local development machines, this is your default profile.
                SessionAWSCredentials tempCredentials =
                    await GetTemporaryFederatedCredentialsAsync();

                // Create a client by providing temporary security credentials.
                using (client = new AmazonS3Client(bucketRegion))
                {
                    ListObjectsRequest listObjectRequest = new
ListObjectsRequest();
                    listObjectRequest.BucketName = bucketName;

                    ListObjectsResponse response = await
client.ListObjectsAsync(listObjectRequest);
                    List<S3Object> objects = response.S3Objects;
                    Console.WriteLine("Object count = {0}", objects.Count);

                    Console.WriteLine("Press any key to continue...");
                    Console.ReadKey();
                }
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}'
when writing an object", e.Message);
            }
        }

        private static async Task<SessionAWSCredentials>
GetTemporaryFederatedCredentialsAsync()
        {
            AmazonSecurityTokenServiceConfig config = new
AmazonSecurityTokenServiceConfig();
            AmazonSecurityTokenServiceClient stsClient =
                new AmazonSecurityTokenServiceClient(
                                            config);
```

```
            GetFederationTokenRequest federationTokenRequest =
                                    new GetFederationTokenRequest();
            federationTokenRequest.DurationSeconds = 7200;
            federationTokenRequest.Name = "User1";
            federationTokenRequest.Policy = @"{
               ""Statement"":
               [
                 {
                   ""Sid"":""Stmt1311212314284"",
                   ""Action"":[""s3:ListBucket""],
                   ""Effect"":""Allow"",
                   ""Resource"":""arn:aws:s3:::" + bucketName + @"""
                  }
               ]
             }
            ";

            GetFederationTokenResponse federationTokenResponse =
                        await
 stsClient.GetFederationTokenAsync(federationTokenRequest);
            Credentials credentials = federationTokenResponse.Credentials;

            SessionAWSCredentials sessionCredentials =
                new SessionAWSCredentials(credentials.AccessKeyId,
                                          credentials.SecretAccessKey,
                                          credentials.SessionToken);
            return sessionCredentials;
        }
    }
}
```

PHP

このトピックでは、バージョン 3 の AWS SDK for PHP のクラスを使用して、フェデレーティッドユーザーおよびアプリケーションの一時的なセキュリティ認証情報をリクエストし、その認証情報を使用して Amazon S3 に保存されているリソースにアクセスする手順を示します。AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

フェデレーティッドユーザーおよびアプリケーションから AWS リソースへのアクセスをリクエストできるように、一時的なセキュリティ認証情報を提供することができます。このような一時的な認証情報をリクエストする場合、ユーザー名と、付与するリソースに対するアクセス許可を

示す IAM ポリシーを指定する必要があります。これらの認証情報は、セッションの有効期間が失効すると同時に失効します。デフォルトでは、セッションの有効期間は 1 時間です。フェデレーティッドユーザーおよびアプリケーション用に一時的なセキュリティ認証情報をリクエストするときに、明示的に有効期間の別の値を設定できます。一時的なセキュリティ認証情報の詳細については、IAM ユーザーガイドの「IAM の一時的なセキュリティ認証情報」を参照してください。フェデレーティッドユーザーおよびアプリケーションに一時的なセキュリティ認証情報を指定する方法については、「リクエストの実行」を参照してください。

フェデレーティッドユーザーおよびアプリケーション用の一時的なセキュリティ認証情報をリクエストする際のセキュリティ強化策として、必要なアクセス許可のみを持つ専用の IAM ユーザーを使用することをお勧めします。作成した一時ユーザーに、一時的なセキュリティ認証情報をリクエストした IAM ユーザーより多くの権限が付与されることはありません。ID フェデレーションについて詳しくは、「AWS Identity and Access Management に関するよくある質問」を参照してください。 。

AWS SDK for Ruby API の詳細については、AWS SDK for Ruby - バージョン 2 を参照してください。

Example

次の PHP の例では、指定したバケットのキーをリストします。この例では、フェデレーティッドユーザー (User1) 用に 1 時間のセッションの一時的なセキュリティ認証情報を取得します。次に、その一時的なセキュリティ認証情報を使用して、認証リクエストを Amazon S3 に送信します。

他のユーザー用に一時的な認証情報をリクエストする場合、セキュリティ強化策として、一時的なセキュリティ認証情報をリクエストするアクセス許可を持つ IAM ユーザーのセキュリティ認証情報を使用します。また、IAM ユーザーがフェデレーティッドユーザーに最小限のアプリケーション固有のアクセス許可しか付与できないように、この IAM ユーザーのアクセス許可を制限することができます。この例では、特定のバケットに含まれるオブジェクトのみ表示されています。以下のポリシーをアタッチして IAM ユーザーを作成します。

```
{
  "Statement":[{
      "Action":["s3:ListBucket",
        "sts:GetFederationToken*"
      ],
      "Effect":"Allow",
      "Resource":"*"
    }
```

```
  ]
}
```

このポリシーでは、IAM ユーザーが一時的なセキュリティ認証情報と、AWS リソースを一覧表示できるだけの許可をリクエストすることを許可しています。IAM ユーザーを作成する方法の詳細については、IAM ユーザーガイドの「最初の IAM ユーザーおよび管理者グループの作成」を参照してください。

これで、IAM ユーザーのセキュリティ認証情報を使用して、以下の例をテストできるようになりました。この例では、一時的なセキュリティ認証情報を使用して、認証リクエストを Amazon S3 に送信します。この例では、フェデレーティッドユーザー (User1) 用に、アクセスを特定のバケット内のオブジェクトの列挙に制限する一時的なセキュリティ認証情報をリクエストするときに、以下のポリシーを指定します。ポリシーは、使用しているバケット名で更新してください。

```
{
  "Statement":[
    {
      "Sid":"1",
      "Action":["s3:ListBucket"],
      "Effect":"Allow",
      "Resource":"arn:aws:s3:::YourBucketName"
    }
  ]
}
```

以下の例では、ポリシーリソースを指定するときに、YourBucketName を、使用しているバケット名で置き換えます。

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

// In real applications, the following code is part of your trusted code. It has
// the security credentials that you use to obtain temporary security credentials.
$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
```

```
]);

// Fetch the federated credentials.
$sessionToken = $sts->getFederationToken([
    'Name'              => 'User1',
    'DurationSeconds'    => '3600',
    'Policy'            => json_encode([
        'Statement' => [
            'Sid'               => 'randomstatementid' . time(),
            'Action'            => ['s3:ListBucket'],
            'Effect'            => 'Allow',
            'Resource'          => 'arn:aws:s3:::' . $bucket
        ]
    ])
]);

// The following will be part of your less trusted code. You provide temporary
// security credentials so the code can send authenticated requests to Amazon S3.

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);

try {
    $result = $s3->listObjects([
        'Bucket' => $bucket
    ]);
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

フェデレーティッドユーザーおよびアプリケーションから AWS リソースへのアクセスのリクエストが認証されて送信されるように、フェデレーティッドユーザーおよびアプリケーションに一時的なセキュリティ認証情報を提供することができます。IAM サービスに対して一時的な認証

情報をリクエストする場合は、ユーザー名と、リソースに対するアクセス許可を付与するための IAM ポリシーを指定する必要があります。デフォルトでは、セッションの有効期間は 1 時間です。ただし、IAM ユーザー認証情報を使用して一時的な認証情報をリクエストする場合、フェデレーションユーザーおよびアプリケーション用の一時的なセキュリティ認証情報をリクエストするときに、別の有効期間値を明示的に設定できます。フェデレーティッドユーザーおよびアプリケーションの一時的なセキュリティ認証情報については、「リクエストの実行」を参照してください。

> **Note**
>
> フェデレーティッドユーザーおよびアプリケーションの一時的なセキュリティ認証情報をリクエストする場合、セキュリティ強化策として、専用の IAM ユーザーを使用して必要なアクセス許可のみを付与できます。作成した一時ユーザーに、一時的なセキュリティ認証情報をリクエストした IAM ユーザーより多くの権限が付与されることはありません。詳細については、「AWS Identity and Access Management に関するよくある質問」を参照してください。

Example

次の Ruby コード例では、フェデレーティッドユーザーに付与するアクセス許可を制限した上で、指定したバケットのキーを一覧表示することを許可します。

```
# Prerequisites:
#  - An existing Amazon S3 bucket.

require "aws-sdk-s3"
require "aws-sdk-iam"
require "json"

# Checks to see whether a user exists in IAM; otherwise,
# creates the user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Aws::IAM::Types::User] The existing or new user.
# @example
#   iam = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam, 'my-user')
#   exit 1 unless user.user_name
#   puts "User's name: #{user.user_name}"
```

```
def get_user(iam, user_name)
  puts "Checking for a user with the name '#{user_name}'..."
  response = iam.get_user(user_name: user_name)
  puts "A user with the name '#{user_name}' already exists."
  return response.user
# If the user doesn't exist, create them.
rescue Aws::IAM::Errors::NoSuchEntity
  puts "A user with the name '#{user_name}' doesn't exist. Creating this user..."
  response = iam.create_user(user_name: user_name)
  iam.wait_until(:user_exists, user_name: user_name)
  puts "Created user with the name '#{user_name}'."
  return response.user
rescue StandardError => e
  puts "Error while accessing or creating the user named '#{user_name}':
 #{e.message}"
end

# Gets temporary AWS credentials for an IAM user with the specified permissions.
#
# @param sts [Aws::STS::Client] An initialized AWS STS client.
# @param duration_seconds [Integer] The number of seconds for valid credentials.
# @param user_name [String] The user's name.
# @param policy [Hash] The access policy.
# @return [Aws::STS::Types::Credentials] AWS credentials for API authentication.
# @example
#   sts = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_temporary_credentials(sts, duration_seconds, user_name,
#     {
#       'Version' => '2012-10-17',
#       'Statement' => [
#         'Sid' => 'Stmt1',
#         'Effect' => 'Allow',
#         'Action' => 's3:ListBucket',
#         'Resource' => 'arn:aws:s3:::doc-example-bucket'
#       ]
#     }
#   )
#   exit 1 unless credentials.access_key_id
#   puts "Access key ID: #{credentials.access_key_id}"
def get_temporary_credentials(sts, duration_seconds, user_name, policy)
  response = sts.get_federation_token(
    duration_seconds: duration_seconds,
    name: user_name,
    policy: policy.to_json
```

```
  )
  return response.credentials
rescue StandardError => e
  puts "Error while getting federation token: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'doc-example-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts "Name => ETag"
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end

# Example usage:
def run_me
  region = "us-west-2"
  user_name = "my-user"
  bucket_name = "doc-example-bucket"

  iam = Aws::IAM::Client.new(region: region)
  user = get_user(iam, user_name)
```

```
  exit 1 unless user.user_name

  puts "User's name: #{user.user_name}"
  sts = Aws::STS::Client.new(region: region)
  credentials = get_temporary_credentials(sts, 3600, user_name,
    {
      "Version" => "2012-10-17",
      "Statement" => [
        "Sid" => "Stmt1",
        "Effect" => "Allow",
        "Action" => "s3:ListBucket",
        "Resource" => "arn:aws:s3:::#{bucket_name}"
      ]
    }
  )

  exit 1 unless credentials.access_key_id

  puts "Access key ID: #{credentials.access_key_id}"
  s3_client = Aws::S3::Client.new(region: region, credentials: credentials)

  exit 1 unless list_objects_in_bucket?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

関連リソース

- AWS SDK を使用した Amazon S3 での開発
- Amazon S3 向け AWS SDK for PHP Aws\S3\S3Client クラスの場合
- AWS SDK for PHP ドキュメント

REST API を使用したリクエストの実行

このセクションでは、REST API を使用して Amazon S3 のエンドポイントにリクエストを生成する方法について説明します。Amazon S3 エンドポイントのリストは、AWS 全般のリファレンス の「リージョンとエンドポイント」を参照してください。

REST API リクエスト用の S3 ホスト名の生成

Amazon S3 エンドポイントは、以下の構造に従います。

```
s3.Region.amazonaws.com
```

Amazon S3 アクセスポイントエンドポイントおよびデュアルスタックエンドポイントも標準構造に従います。

- Amazon S3 アクセスポイント -s3-accesspoint.*Region*.amazonaws.com
- デュアルスタック - s3.dualstack.*Region*.amazonaws.com

Amazon S3 リージョンとエンドポイントの完全なリストについては、Amazon Web Services 全般のリファレンス の「Amazon S3 エンドポイントとクォータ」を参照してください。

仮想ホスティング形式およびパス形式のリクエスト

REST API を使用してリクエストを生成すると、Amazon S3 のエンドポイントに仮想ホスティング形式またはパス形式の URI を使用できます。詳細については、「バケットの仮想ホスティング」を参照してください。

Example 仮想ホスティング形式のリクエスト

米国西部 (オレゴン) リージョンにある puppy.jpg ファイルを examplebucket という名前のバケットから削除する仮想ホスティング形式のリクエストの例を次に示します。仮想ホスト形式のリクエストの詳細については、「仮想ホスト形式のリクエスト」を参照してください。

```
DELETE /puppy.jpg HTTP/1.1
Host: examplebucket.s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Example パス形式のリクエスト

同じリクエストのパス形式バージョンの例を次に示します。

```
DELETE /examplebucket/puppy.jpg HTTP/1.1
```

```
Host: s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

現在、Amazon S3 では、すべての AWS リージョン で仮想ホスト形式の URL とパス形式の URL の両方をサポートしています。ただし、パス形式の URL は将来廃止される予定です。詳細については、次の重要な注記を参照してください。

パス形式のリクエストの詳細については、「パス形式のリクエスト」を参照してください。

> **Important**
>
> 更新 (2020 年 9 月 23 日) – お客様が仮想ホスティング形式の URL への移行に必要な時間を確保できるように、パス形式 URL の非推奨化を延期することが決定しました。詳細については、AWS ニュースブログの Amazon S3 Path Deprecation Plan – The Rest of the Story を参照してください。

REST API を使用したデュアルスタックのエンドポイントへのリクエストの実行

REST API を使用すると、仮想ホスティング形式やパス形式のエンドポイント名 (URI) を使用してデュアルスタックのエンドポイントに直接アクセスできます。すべての Amazon S3 デュアルスタックのエンドポイント名は名前にリージョンが含まれます。標準の IPv4 のみのエンドポイントとは異なり、仮想ホスティング形式とパス形式のエンドポイントではリージョン固有のエンドポイント名を使用します。

Example 仮想ホスティング形式のデュアルスタックのエンドポイントのリクエスト

米国西部 (オレゴン) リージョンにある `puppy.jpg` という名前のバケットから `examplebucket` オブジェクトを取得する次の例に示すように、REST リクエストで仮想ホスティング形式のエンドポイントを使用できます。

```
GET /puppy.jpg HTTP/1.1
Host: examplebucket.s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Example パス形式のデュアルスタックのエンドポイントのリクエスト

または次の例に示すように、リクエストでパス形式のエンドポイントを使用することができます。

```
GET /examplebucket/puppy.jpg HTTP/1.1
Host: s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

デュアルスタックのエンドポイントの詳細については、「Amazon S3 デュアルスタックのエンドポイントの使用」を参照してください。

REST API を使用したリクエストの詳細については、以下のトピックをご参照ください。

トピック

- バケットの仮想ホスティング
- リクエストのリダイレクトと REST API

バケットの仮想ホスティング

仮想ホスティングとは、単一のウェブサーバーから複数のウェブサイトにサービスを提供することです。Amazon S3 REST API リクエストでサイトを区別する方法の 1 つとして、単なる URI のパス名部分ではなく、リクエスト URI の明確なホスト名を使用します。通常の Amazon S3 REST リクエストは、リクエスト URI パスのスラッシュで区切られた先頭コンポーネントを使用してバケットを指定します。代わりに、Amazon S3 仮想ホスティングでは、HTTP `Host` ヘッダーを使用することで、REST API コールでバケットを指定することができます。実際には、Amazon S3 は `Host` を、ほとんどのバケットは `https://bucket-name.s3.region-code.amazonaws.com` で自動的にアクセス可能である意味であると解釈します (限定されたタイプのリクエストの場合)。Amazon S3 リージョンとエンドポイントの完全なリストについては、Amazon Web Services 全般のリファレンス の「Amazon S3 エンドポイントとクォータ」を参照してください。

仮想ホスティングには、他の利点もあります。さらに、登録されたドメイン名を使用してバケットの名前を指定し、その名前を Amazon S3 の DNS エイリアスにすることによって、Amazon S3 リソースの URL を完全にカスタマイズすることができます (例: `http://my.bucket-name.com/`)。バケットの仮想サーバーの「ルートディレクトリ」に公開することもできます。多くの既存のアプリケーションがこの標準ロケーションでファイルを検索するため、この機能は重要です。たとえば、`favicon.ico`、`robots.txt`、および `crossdomain.xml` はすべてルートで見つかることが見込まれています。

> **Important**
>
> SSL と共に仮想ホスト形式のバケットを使用している場合、SSL ワイルドカード証明書は、ドット (「.」) を含まないバケットのみと一致します。この制限を回避するには、HTTP を使用するか、または独自の証明書検証ロジックを記述します。詳細については、AWS ニュースブログのAmazon S3 パスの非推奨化プランを参照してください。

トピック

パス形式のリクエスト

現在、Amazon S3 では、すべての AWS リージョン で仮想ホスト形式の URL とパス形式の URL の両方をサポートしています。ただし、パス形式の URL は将来廃止される予定です。詳細については、次の重要な注記を参照してください。

Amazon S3 では、パス形式の URL は次の形式を使用します。

```
https://s3.region-code.amazonaws.com/bucket-name/key-name
```

例えば、DOC-EXAMPLE-BUCKET1 という名前のバケットを 米国西部 (オレゴン) リージョンで作成し、そのバケットで puppy.jpg オブジェクトにアクセスする場合、次のパス形式の URL を使用できます。

```
https://s3.us-west-2.amazonaws.com/DOC-EXAMPLE-BUCKET1/puppy.jpg
```

> **Important**
>
> 更新 (2020 年 9 月 23 日) – お客様が仮想ホスティング形式の URL への移行に必要な時間を確保できるように、パス形式 URL の非推奨化を延期することが決定しました。詳細については、AWS ニュースブログ の「Amazon S3 Path Deprecation Plan – The Rest of the Story」を参照してください。

> **Warning**
>
> ウェブブラウザからアクセスするウェブサイトのコンテンツをホストする場合は、生成元が同じブラウザのセキュリティモデルを妨げる可能性があるパススタイルの URL の使用を避けてください。ウェブサイトコンテンツをホストするには、S3 ウェブサイトエンドポイントまたは CloudFront ディストリビューションのいずれかを使用することをお勧めします。詳細については、「ウェブサイトエンドポイント」およびAWS 規範的ガイダンスパターンの「Deploy a React-based single-page application to Amazon S3 and CloudFront」(React ベースの単一ページアプリケーションを Amazon S3 と CloudFront にデプロイする) を参照してください。

仮想ホスト形式のリクエスト

仮想ホスティング形式の URI では、バケット名は URL のドメイン名の一部です。

Amazon S3 仮想ホスト形式の URL は、次の形式を使用します。

```
https://bucket-name.s3.region-code.amazonaws.com/key-name
```

この例では、DOC-EXAMPLE-BUCKET1 はバケット名、米国西部 (オレゴン) はリージョン、puppy.png はキー名です。

```
https://DOC-EXAMPLE-BUCKET1.s3.us-west-2.amazonaws.com/puppy.png
```

HTTP **Host** ヘッダーバケット仕様

GET リクエストが SSL エンドポイントを使用しないかぎり、HTTP Host ヘッダーを使用してリクエストに対してバケットを指定できます。REST リクエストの Host ヘッダーは、次のように解釈されます。

- Host ヘッダーが省略されるか、またはその値が s3.*region-code*.amazonaws.com である場合、リクエストのバケットはリクエスト URI のスラッシュで区切られた先頭コンポーネント、リクエストのキーはリクエスト URI の残りの部分になります。このセクションの最初および 2 つ目の例に示すように、これが通常の方式です。Host ヘッダーは、HTTP 1.0 リクエストでのみ省略可能です。
- 上記のいずれの条件も該当せず、Host ヘッダーの値が .s3.*region-code*.amazonaws.com で終わる場合、バケット名は Host ヘッダーの値のうち、先頭から .s3.*region-code*.amazonaws.com までの値になります。リクエストのキーはリクエスト URI になります。この解釈により、このセクションの 3 番目と 4 番目の例で示されているように、.s3.*region-code*.amazonaws.com のサブドメインとしてバケットが公開されます。
- それ以外の場合、リクエストのバケットは Host ヘッダーの小文字の値、リクエストのキーはリクエスト URI になります。この解釈は、バケット名と同じ DNS 名を登録してして、その名前を Amazon S3 の 正規名 (CNAME) エイリアスとして設定している場合に有用です。このドキュメントではドメイン名の登録および CNAME DNS レコードの設定の手順については扱いませんが、このセクションの最後の例にその結果を示します。

例

このセクションでは、URL およびリクエストの例を示します。

Example — パス形式の URL とリクエスト

この例では以下を使用しています。

- バケット名 - example.com
- リージョン - 米国東部 (バージニア北部)
- キー名 - homepage.html

URL は次のとおりです。

```
http://s3.us-east-1.amazonaws.com/example.com/homepage.html
```

リクエストは次のとおりです。

```
GET /example.com/homepage.html HTTP/1.1
Host: s3.us-east-1.amazonaws.com
```

HTTP 1.0 でのリクエストと Host ヘッダーの省略は次のとおりです。

```
GET /example.com/homepage.html HTTP/1.0
```

DNS 互換名については、「制約事項」を参照してください。キーの詳細については、「キー」を参照してください。

Example — 仮想ホスト形式の URL とリクエスト

この例では以下を使用しています。

- バケット名 - DOC-EXAMPLE-BUCKET1
- リージョン - 欧州 (アイルランド)
- キー名 - homepage.html

URL は次のとおりです。

```
http://DOC-EXAMPLE-BUCKET1.s3.eu-west-1.amazonaws.com/homepage.html
```

リクエストは次のとおりです。

```
GET /homepage.html HTTP/1.1
Host: DOC-EXAMPLE-BUCKET1.s3.eu-west-1.amazonaws.com
```

Example — CNAME エイリアス方式

この方式を使用するには、DNS 名を *bucket-name*.s3.us-east-1.amazonaws.com の CNAME エイリアスとして設定する必要があります。詳細については、「CNAME レコードを使用した Amazon S3 URL のカスタマイズ」を参照してください。

この例では以下を使用しています。

- バケット名 - example.com
- キー名 - homepage.html

URL は次のとおりです。

```
http://www.example.com/homepage.html
```

例は次のとおりです。

```
GET /homepage.html HTTP/1.1
Host: www.example.com
```

CNAME レコードを使用した Amazon S3 URL のカスタマイズ

要件によっては、「s3.*region-code*.amazonaws.com」をウェブサイトまたはサービスに表示したくない場合もあります。例えば、ウェブサイトの画像を Amazon S3 でホスティングしている場合、http://images.example.com/ の代わりに http://images.example.com.s3.us-east-1.amazonaws.com/ と表示することができます。DNS 互換名を持つすべてのバケットは、http://*BucketName*.s3.*Region*.amazonaws.com/[*Filename*] のように参照できます (例: http://images.example.com.s3.us-east-1.amazonaws.com/mydog.jpg)。CNAME を使用すると、images.example.com を Amazon S3 のホスト名にマッピングできるため、この URL は http://images.example.com/mydog.jpg になります。

バケット名は CNAME と同じである必要があります。例えば、CNAME を作成して images.example.com を images.example.com.s3.us-east-1.amazonaws.com にマッピングした場合、http://images.example.com/filename と http://images.example.com.s3.us-east-1.amazonaws.com/filename はどちらも同じになります。

CNAME DNS レコードは、ドメイン名を適切な仮想ホスティング形式のホスト名にエイリアス設定する必要があります。たとえば、バケット名とドメイン名が images.example.com で、バケットが 米国東部 (バージニア北部) リージョンにある場合、CNAME レコードは images.example.com.s3.us-east-1.amazonaws.com のエイリアスであることが必要です。

```
images.example.com CNAME    images.example.com.s3.us-east-1.amazonaws.com.
```

Amazon S3 はホスト名を使用して、バケット名を決定します。そのため CNAME とバケット名は同じである必要があります。たとえば、www.example.com を www.example.com.s3.us-east-1.amazonaws.com の CNAME として設定したとします。http://www.example.com にアクセスすると、Amazon S3 は次のようなリクエストを受け取ります。

Example

```
GET / HTTP/1.1
Host: www.example.com
```

```
Date: date
Authorization: signatureValue
```

Amazon S3 は元のホスト名 www.example.com だけを認識し、リクエストを解決するために使用される CNAME マッピングを認識しません。

どの Amazon S3 エンドポイントも、CNAME エイリアスで使用できます。たとえば、s3.ap-southeast-1.amazonaws.com は CNAME エイリアスで使用できます。エンドポイントの詳細については、「エンドポイントのリクエスト」を参照してください。カスタムドメインを使用して静的ウェブサイトを作成するには、チュートリアル: Route 53 に登録されたカスタムドメインを使用した静的ウェブサイトの設定 を参照してください。

> **Important**
>
> CNAME でカスタム URL を使用する場合は、設定した CNAME またはエイリアスレコードに一致するバケットが存在することを確認する必要があります。たとえば、www.example.com と login.example.com の DNS エントリを作成し、S3 を使用してウェブコンテンツを公開するには、www.example.com と login.example.com の両方のバケットを作成する必要があります。
> CNAME またはエイリアスレコードが、一致するバケットがない S3 エンドポイントをポイントするように設定されている場合、AWS ユーザーは、所有権が同じでなくても、そのバケットを作成し、設定されたエイリアスでコンテンツを公開できます。
> 同じ理由で、バケットを削除するときは、対応する CNAME またはエイリアスを変更または削除することをお勧めします。

ホスト名を Amazon S3 バケットに関連付ける方法

CNAME エイリアスを使用してホスト名を Amazon S3 バケットに関連付けるには

1. 管理するドメインに属するホスト名を選択します。

   この例では、images ドメインの example.com サブドメインを使用します。
2. ホスト名と一致するバケットを作成します。

   この例では、ホスト名およびバケット名は images.example.com です。バケット名はホスト名と完全に一致する必要があります。
3. ホスト名を Amazon S3 バケットのエイリアスとして定義する CNAME DNS レコードを作成します。

例:

```
images.example.com CNAME images.example.com.s3.us-west-2.amazonaws.com
```

> **Important**
>
> リクエストルーティングの理由により、CNAME DNS レコードは、前述の例と完全に一致するように定義する必要があります。そうしない場合、正しく動作するように見えても、最終的には予期しない結果を招くことがあります。

CNAME DNS レコードを設定する手順は、お使いの DNS サーバーまたは DNS プロバイダーによって異なります。個別の情報については、お使いのサーバーのマニュアルをご覧になるか、プロバイダーにお問い合わせください。

制限事項

SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

下位互換性

以下のセクションでは、パス形式および仮想ホスト形式の URL リクエストに関連する Amazon S3 の下位互換性のさまざまな側面について説明します。

レガシーエンドポイント

一部のリージョンでは、レガシーエンドポイントがサポートされています。これらのエンドポイントは、サーバーアクセスログまたは AWS CloudTrail ログに表示される場合があります。詳細については、次の情報を確認してください。Amazon S3 リージョンとエンドポイントの完全なリストについては、Amazon Web Services 全般のリファレンス の「Amazon S3 エンドポイントとクォータ」を参照してください。

> **Important**
>
> ログにレガシーエンドポイントが表示される場合もありますが、バケットにアクセスするには、常に標準のエンドポイント構文を使用することをお勧めします。

Amazon S3 仮想ホスト形式の URL は、次の形式を使用します。

```
https://bucket-name.s3.region-code.amazonaws.com/key-name
```

Amazon S3 では、パス形式の URL は次の形式を使用します。

```
https://s3.region-code.amazonaws.com/bucket-name/key-name
```

s3-Region

一部の古い Amazon S3 リージョンでは、ドット (s3.us-west-2 など) ではなく、s3 とリージョンコードの間にダッシュ (-) を含むエンドポイント (s3-us-west-2 など) がサポートされています。バケットがこれらのリージョンのいずれかにある場合、サーバーアクセスログまたは CloudTrail ログに次のエンドポイント形式が表示されることがあります。

```
https://bucket-name.s3-region-code.amazonaws.com
```

この例では、バケット名は DOC-EXAMPLE-BUCKET1 で、リージョンは米国西部 (オレゴン) です。

```
https://DOC-EXAMPLE-BUCKET1.s3-us-west-2.amazonaws.com
```

レガシーグローバルエンドポイント

一部のリージョンでは、レガシーグローバルエンドポイントを使用して、リージョン固有のエンドポイントを指定しないリクエストを作成できます。レガシーグローバルエンドポイントポイントは次のとおりです。

```
bucket-name.s3.amazonaws.com
```

サーバーアクセスログまたは CloudTrail ログに、レガシーグローバルエンドポイントを使用するリクエストが表示される場合があります。この例では、バケット名は DOC-EXAMPLE-BUCKET1 であり、レガシーグローバルエンドポイントは次のとおりです。

```
https://DOC-EXAMPLE-BUCKET1.s3.amazonaws.com
```

米国東部 (バージニア北部) の仮想ホスト形式のリクエスト

レガシーグローバルエンドポイントで行われたリクエストは、デフォルトでは米国東部 (バージニア北部) リージョンに送信されます。したがって、レガシーグローバルエンドポイントは、米国東部 (バージニア北部) のリージョン別エンドポイントの代わりに使用されることがあります。米国東部 (バージニア北部) にバケットを作成し、グローバルエンドポイントを使用する場合、Amazon S3 はデフォルトでリクエストをこのリージョンにルーティングします。

他のリージョンに対する仮想ホスト形式のリクエスト

レガシーグローバルエンドポイントは、サポートされている他のリージョンでの仮想ホスト形式のリクエストにも使用されます。2019 年 3 月 20 日より前に開始されたリージョンにバケットを作成し、レガシーグローバルエンドポイントを使用すると、Amazon S3 は DNS レコードを更新してリクエストを正しいロケーションにルーティングします。この処理には時間がかかる場合があります。その間、デフォルトのルールが適用され、仮想ホスト形式のリクエストは米国東部 (バージニア北部) リージョンに送られます。次に、Amazon S3 は HTTP 307 一時的リダイレクトを使用して正しいリージョンにリダイレクトします。

2019 年 3 月 20 日より後に開始されたリージョンの S3 バケットの場合、DNS サーバーはリクエストをバケットが存在する AWS リージョン に直接ルーティングしません。代わりに HTTP 400 Bad Request エラーが返されます。詳細については、「リクエストの実行」を参照してください。

パス形式のリクエスト

米国東部 (バージニア北部) リージョンでは、パス形式のリクエストにレガシーグローバルエンドポイントを使用できます。

他のすべてのリージョンでは、パススタイル構文の場合、バケットへのアクセスを試行するときにリージョン固有のエンドポイントを使用する必要があります。レガシーグローバルエンドポイント、またはバケットが存在するリージョンのエンドポイントとは異なる別のエンドポイントを使用してバケットにアクセスしようとすると、HTTP レスポンスコード 307「一時的なリダイレクト」エラーと、リソースの正しい URI を示すメッセージが表示されます。たとえば、米国西部 (オレゴン) リージョンで作成されたバケットに `https://s3.amazonaws.com/bucket-name` を使用すると、HTTP 307 Temporary Redirect エラーが発生します。

リクエストのリダイレクトと REST API

トピック

- リダイレクトおよび HTTP ユーザーエージェント
- リダイレクトおよび 100-continue

- リダイレクトの例

このセクションでは、Amazon S3 REST API を使用して HTTP リダイレクトを処理する方法について説明します。Amazon S3 リダイレクトの一般的な情報については、Amazon Simple Storage Service API リファレンスの「リクエストの実行」を参照してください。

リダイレクトおよび HTTP ユーザーエージェント

Amazon S3 REST API を使用するプログラムは、アプリケーションレイヤーレベルまたは HTTP レイヤーレベルのいずれかでリダイレクトを処理する必要があります。HTTP クライアントライブラリとユーザーエージェントの多くは、リダイレクトを自動的に正しく処理するよう設定することができます。ただし、リダイレクト実装が不正確あるいは不完全であるものも多くあります。

ライブラリでリダイレクト要件を満たすようにする前に、次のテストを行ってください。

- すべての HTTP リクエストヘッダーが (Authorization、Date などの HTTP 標準を含む)、リダイレクトされたリクエスト (リダイレクトの受信後の 2 番目のリクエスト) に適切に含まれていることを確認します。
- PUT、DELETE など、非 GET リダイレクトが正しく動作することを確認します。
- 大きな PUT リクエストがリダイレクトに正しく従っていることを確認します。
- 100-continue レスポンスが届くまでに時間がかかる場合は、PUT リクエストがリダイレクトに正しく従っていることを確認します。

HTTP リクエストが GET または HEAD でない場合、RFC 2616 に厳密に準拠する HTTP ユーザーエージェントは、リダイレクトにしたがう前に明示的な確認が必要になることがあります。一般的に、Amazon S3 によって自動生成されたリダイレクトに従うと安全です。システムが amazonaws.com ドメイン内のホストのみに対してリダイレクトを発行し、リダイレクトされたリクエストの効果が元のリクエストと同じになるためです。

リダイレクトおよび 100-continue

リダイレクトの処理を簡素化し、効率性を高め、リダイレクトされたリクエストボディを 2 回送信するためのコストを排除するには、PUT オペレーションに対して 100-continue を使用するようにアプリケーションを設定します。100-continue を使用しているアプリケーションは、確認を受け取るまでリクエストボディを送信しません。ヘッダーに基づいてメッセージが拒否された場合、メッセージの本文は送信されません。100-continue の詳細については、「RFC 2616 Section 8.2.3」を参照してください。

> **Note**
>
> RFC 2616 にしたがって、不明な HTTP サーバーで `Expect: Continue` を使用する場合は、リクエストボディを送信する前の待機時間は無期限にしないでください。HTTP サーバーの中には、100-continue を認識しないものがあるからです。しかし、Amazon S3 は、リクエストに `Expect: Continue` が含まれているかどうか、また一時的な 100-continue ステータスを返すか、最終ステータスコードを返すかを認識します。さらに、一時的な 100 continue の承認を受け取った後、リダイレクトエラーが発生することはありません。これは、リクエスト本文を書き込んでいる間は、リダイレクトレスポンスを受け取らないようにする際に役立ちます。

リダイレクトの例

このセクションでは、HTTP リダイレクトと 100-continue を使用したクライアントとサーバーのやり取りの例を紹介します。

`quotes.s3.amazonaws.com` バケットに対する PUT の例を次に示します。

```
PUT /nelson.txt HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 は以下を返します。

```
HTTP/1.1 307 Temporary Redirect
Location: http://quotes.s3-4c25d83b.amazonaws.com/nelson.txt?rk=8d47490b
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Mon, 15 Oct 2007 22:18:46 GMT

Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
```

```
  <Message>Please re-send this request to the
  specified temporary endpoint. Continue to use the
  original request endpoint for future requests.
  </Message>
  <Endpoint>quotes.s3-4c25d83b.amazonaws.com</Endpoint>
  <Bucket>quotes</Bucket>
</Error>
```

クライアントは、リダイレクトレスポンスにしたがって、新しいリクエストを quotes.s3-4c25d83b.amazonaws.com 一時エンドポイントに発行します。

```
PUT /nelson.txt?rk=8d47490b HTTP/1.1
Host: quotes.s3-4c25d83b.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 は 100-continue 返します。これは、クライアントがリクエストボディを送信し続ける必要があることを示します。

```
HTTP/1.1 100 Continue
```

クライアントはリクエストボディを送信します。

```
ha ha\n
```

Amazon S3 は最終レスポンスを返します。

```
HTTP/1.1 200 OK
Date: Mon, 15 Oct 2007 22:18:48 GMT

ETag: "a2c8d6b872054293afd41061e93bc289"
Content-Length: 0
Server: AmazonS3
```

AWS CLI を使用した Amazon S3 での開発

AWS Command Line Interface (AWS CLI) をダウンロードして設定するには、次の手順に従います。

Amazon S3 AWS CLI コマンドの一覧については、AWS CLI コマンドリファレンスの次のページを参照してください。

- s3
- s3api
- s3control

> **Note**
>
> Amazon S3 などの AWS のサービスは、サービスにアクセスする際に認証情報を提供する必要があります。これにより、サービスのリソースへアクセスするための権限があるかどうかが判断できるようになります。コンソールを使用するにはパスワードが必要です。AWS アカウント による AWS CLI または API へのアクセス用にアクセスキーを作成できます。ただし、AWS アカウント の認証情報を使用して AWS にアクセスすることは推奨されません。代わりに AWS Identity and Access Management (IAM) を使用することをお勧めします。IAM ユーザーを作成し、管理者アクセス許可を持つ IAM グループにユーザーを追加したら、作成した IAM ユーザーに管理者アクセス許可を付与します。これで、特別な URL と IAM ユーザーの認証情報を使って AWS にアクセスできます。手順については、IAM ユーザーガイドの「最初の IAM ユーザーと管理者グループの作成」を参照してください。

AWS CLI をセットアップするには

1. AWS CLI をダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。
   - AWS Command Line Interface のセットアップ
   - AWS Command Line Interface の設定
2. AWS CLI 設定ファイルで管理者ユーザー用の名前付きプロファイルを追加します。AWS CLI コマンドを実行するときに、このプロファイルを使用します。詳細については、AWS Command Line Interface ユーザーガイド の「AWS CLI の名前付きプロファイル」を参照してください。

```
[adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

使用可能な AWS リージョン のリストについては、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。

3. コマンドプロンプトで以下のコマンドを入力して、セットアップを確認します。

   - help コマンドを実行して、AWS CLI がコンピュータにインストールされたことを確認します。

     ```
     aws help
     ```

   - 作成した adminuser 認証情報を使用して、S3 コマンドを実行します。これを行うには、--profile パラメータをコマンドに追加し、プロファイル名を指定します。この例では、ls コマンドにより、アカウント内のバケットがリストされます。AWS CLI は、adminuser 認証情報を使用してリクエストを認証します。

     ```
     aws s3 ls --profile adminuser
     ```

AWS SDK を使用した Amazon S3 での開発

AWS ソフトウェア開発キット (SDK) は、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

> **Note**
>
> AWS Amplify を使用して、ウェブおよびモバイルアプリのエンドツーエンドのフルスタック開発を行うことができます。Amplify Storage は、Amazon S3 上に構築されたフロントエンドのウェブおよびモバイルアプリにファイルストレージと管理機能をシームレスに統合します。詳細については、「Amplify ユーザーガイド」の「Storage」を参照してください。

このサービスを AWS SDK で使用する

AWS ソフトウェア開発キット (SDK) は、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コードの例
AWS SDK for C++	AWS SDK for C++ コードの例
AWS CLI	AWS CLI コードの例
AWS SDK for Go	AWS SDK for Go コードの例
AWS SDK for Java	AWS SDK for Java コードの例
AWS SDK for JavaScript	AWS SDK for JavaScript コードの例
AWS SDK for Kotlin	AWS SDK for Kotlin コードの例
AWS SDK for .NET	AWS SDK for .NET コードの例
AWS SDK for PHP	AWS SDK for PHP コードの例
AWS Tools for PowerShell	Tools for PowerShell のコード例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コードの例
AWS SDK for Ruby	AWS SDK for Ruby コードの例
AWS SDK for Rust	AWS SDK for Rust コードの例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コードの例
AWS SDK for Swift	AWS SDK for Swift コードの例

このサービスに固有の例については、「AWS SDK を使用した Amazon S3 のコード例」を参照してください。

> **可用性の例**
>
> 必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

SDK プログラミングインターフェイス

各 AWS SDK には、Amazon S3 を操作するプログラミングインターフェイスが 1 つ以上用意されています。各 SDK は、Amazon S3 用の低レベルインターフェイスを提供します。これには、API オペレーションに似たメソッドが含まれます。一部の SDK は、一般的なユースケースを簡素化するための抽象化である Amazon S3 用の高レベルインターフェイスを提供します。

例えば、低レベル API オペレーションを使用してマルチパートアップロードを実行する場合は、オペレーションを使用してアップロードを開始し、別のオペレーションを使用してパートをアップロードし、最後のオペレーションを使用してアップロードを完了する必要があります。高レベルのマルチパートアップロード API オペレーションを使用すると、1 回の API コールでアップロードに必要なすべてのオペレーションを実行できます。例については、「マルチパートアップロードを使用したオブジェクトのアップロード」を参照してください。

低レベルの API オペレーションを使用すると、アップロードをより詳細に制御できます。アップロードを一時停止して再開する必要がある場合、アップロード中にパートサイズを変更する場合、またはアップロードデータのサイズが事前にわからない場合は、低レベル API オペレーションの使用をお勧めします。

リクエスト認証での署名バージョンの指定

Amazon S3 は、ほとんどの AWS リージョン で AWS Signature Version 4 のみをサポートしています。ただし、初期の AWS リージョン のいくつかでは、Amazon S3 は Signature Version 4 と Signature Version 2 をサポートしています。ただし、署名バージョン 2 はオフになっています (廃止)。署名バージョン 2 のサポート終了についての詳細は、「AWSAmazon S3 の 署名バージョン 2 がオフになっている (非推奨)」を参照してください。

すべての Amazon S3 リージョンとリージョンでサポートされている署名バージョンのリストについては AWS 全般のリファレンスのリージョンとエンドポイントを参照してください。

すべての AWS リージョン では、AWS SDK はデフォルトで Signature Version 4 を使用してリクエストを認証します。2016 年 5 月以前にリリースされた AWS SDK を使用する場合、次の表に示すように、Signature Version 4 のリクエストが必要になることがあります。

SDK	リクエスト認証に署名バージョン 4 を使用するようリクエストする
AWS CLI	デフォルトプロファイルの場合、次のコマンドを実行します。

SDK	リクエスト認証に署名バージョン 4 を使用するようリクエストする
	`$ aws configure set default.s3.signature_version s3v4` カスタムプロファイルの場合、次のコマンドを実行します。 `$ aws configure set profile.your_profile_name.s3.signature_version s3v4`
Java SDK	コードに以下を追加します: `System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");` または、コマンドラインで次のように指定します。 `-Dcom.amazonaws.services.s3.enableV4`
JavaScript SDK	クライアントを構築するときに、`signatureVersion` パラメータを `v4` に設定します。 `var s3 = new AWS.S3({signatureVersion: 'v4'});`

SDK	リクエスト認証に署名バージョン 4 を使用するようリクエストする
PHP SDK	PHP SDK v2 用の Amazon S3 サービスクライアントを構築する場合は、`signature` パラメータを v4 に設定します。 `<?php` `$client = S3Client::factory([` `'region' => 'YOUR-REGION',` `'version' => 'latest',` `'signature' => 'v4'` `]);` PHP SDK v3 を使用している場合、Amazon S3 サービスクライアントの構築中に `signature_version` パラメータを v4 に設定します。 `<?php` `$s3 = new Aws\S3\S3Client([` `'version' => '2006-03-01',` `'region' => 'YOUR-REGION',` `'signature_version' => 'v4'` `]);`
Python-Boto SDK	boto デフォルト設定ファイルに以下を指定します。 `[s3] use-sigv4 = True`

SDK	リクエスト認証に署名バージョン 4 を使用するようリクエストする
Ruby SDK	Ruby SDK - バージョン 1: クライアントを構築するときに :s3_signature_version パラメータを :v4 に設定します。 `s3 = AWS::S3::Client.new(:s3_signature_version => :v4)` Ruby SDK - バージョン 3: クライアントを構築するときに signature_version パラメータを v4 に設定します。 `s3 = Aws::S3::Client.new(signature_version: 'v4')`
.NET SDK	Amazon S3 クライアントを作成する前に、コードに以下を追加します。 `AWSConfigsS3.UseSignatureVersion4 = true;` または、以下を設定ファイルに追加します: `<appSettings>` `  <add key="AWS.S3.UseSignatureVersion4" value="true" />` `</appSettings>`

AWSAmazon S3 の 署名バージョン 2 がオフになっている (非推奨)

署名バージョン 2 は Amazon S3 でオフになっています (廃止)。Amazon S3 は署名バージョン 4 を使用して署名された API リクエストのみを受けつけます。

このセクションでは、署名バージョン 2 のサポート終了に関するよくある質問への回答が掲載されています。

署名バージョン 2 または 4 とは何ですか。また、署名リクエストとはどういう意味ですか。

署名バージョン 2 または署名バージョン 4 の署名プロセスは、Amazon S3 API リクエストを認証するために使用されます。リクエストの署名により、Amazon S3 によってリクエストの送信者を識別し、悪意のある人物からリクエストを保護することができます。

AWS リクエストの署名の詳細については、「AWS 全般のリファレンス」の「AWS API リクエストの署名」を参照してください。

どの更新を実行していますか。

現在、署名バージョン 2 および署名バージョン 4 のプロセスを使用して署名された Amazon S3 API リクエストがサポートされています。今後、Amazon S3 は署名バージョン 4 を使用して署名したリクエストのみを受け入れるようになります。

AWS リクエストの署名の詳細については、「AWS 全般のリファレンス」の「Signature Version 4 の変更」を参照してください。

なぜこの更新を実行するのですか。

署名バージョン 4 は、シークレットアクセスキーに代わって、署名されたキーを使用することでセキュリティの向上を提供します。Signature Version 4 は現在すべての AWS リージョン でサポートされていますが、Signature Version 2 は 2014 年 1 月以前にリリースしたリージョンにおいてのみサポートされます。この更新によって、すべてのリージョンを通してより一貫したエクスペリエンスを提供することができます。

署名バージョン 4 を使用していることはどのように確認できますか。また、どの更新が必要ですか。

リクエストを署名するために使用する署名バージョンは、通常の場合、クライアント側のツールあるいは SDK によって設定されます。デフォルトでは、最新バージョンの AWS SDK は Signature Version 4 を使用します。サードパーティのソフトウェアについては、お使いのソフトウェアに該当するサポートチームにお問い合わせいただき、必要なバージョンを確認してください。Amazon S3 にダイレクト REST コールを送信する場合、アプリケーションを変更して署名バージョン 4 の署名プロセスを使用する必要があります。

Signature Version 4 に移行するときに使用する AWS SDK のバージョンについての詳細は、署名バージョン 2 から署名バージョン 4 への移行 を参照してください。

Amazon S3 REST API で Signature Version 4 を使用する方法については、Amazon Simple Storage Service API リファレンスのリクエストの認証 (AWSSignature Version 4) を参照してください。

更新しない場合には、どうなりますか。

以降に署名バージョン 2 で署名されたリクエストは、Amazon S3 における認証で失敗します。リクエスト送信者には、リクエストが署名バージョン 4 で署名されている必要があることのエラーが表示されます。

7 日以上にわたって署名することが必要である署名付き URL を使用していても、変更することが必要ですか。

7 日以上にわたって署名することが必要である署名付き URL を使用している場合、現在のところ対応する必要はありません。引き続き AWS Signature Version 2 を使用して署名し、署名付き URL を認証できます。今後、署名付き URL シナリオにおける署名バージョン 4 への移行方法に関する詳細をフォローアップして、提供する予定です。

詳細

- Signature Version 4 の使用方法の詳細については、AWS API リクエストの署名を参照してください。
- 署名バージョン 2 と 署名バージョン 4 の変更点のリストについては、「署名バージョン 4 の変更」を参照してください。
- AWS フォーラムの投稿である Amazon S3 API リクエストの署名における AWS Signature Version 2 から AWS Signature Version 4 への移行を参照してください。
- ご質問やご意見がありましたら、AWS Support までお問い合わせください。

署名バージョン 2 から署名バージョン 4 への移行

Amazon S3 API リクエスト認証に署名バージョン 2 を現在使用している場合は、署名バージョン 4 の使用に移行する必要があります。「AWSAmazon S3 の 署名バージョン 2 がオフになっている (非推奨)」で説明されているとおり、署名バージョン 2 のサポートは終了します。

Amazon S3 REST API で Signature Version 4 を使用する方法については、Amazon Simple Storage Service API リファレンスのリクエストの認証 (AWSSignature Version 4) を参照してください。

次の表では、署名バージョン 4 (SigV4) を使用するために必要な最低限のバージョンの SDK の一覧を示しています。AWS Java、JavaScript (Node.js)、あるいは Python (Boto/CLI) SDK を使った署名付き URL を使用している場合は、正しい AWS リージョン を設定し、クライアント設定で Signature Version 4 を設定する必要があります。クライアント設定で `SigV4` を設定する詳細については、「リクエスト認証での署名バージョンの指定」を参照してください。

この SDK または製品を使用する場合	この SDK バージョンにアップグレード	Sigv4 を使用するために、クライアントでコード変更が必要ですか。	SDK ドキュメントへのリンク
AWS SDK for Java v1	Java 1.11.201 以降または v2 にアップグレード。	あり	リクエスト認証での署名バージョンの指定
AWS SDK for Java v2	SDK のアップグレードは不要です。	なし	AWS SDK for Java
AWS SDK for .NET v1	3.1.10 以降にアップグレード。	あり	AWS SDK for .NET
AWS SDK for .NET v2	3.1.10 以降にアップグレード。	なし	AWS SDK for .NET v2
AWS SDK for .NET v3	3.3.0.0 以降にアップグレード。	あり	AWS SDK for .NET v3
AWS SDK for JavaScript v1	2.68.0 以降にアップグレード。	あり	AWS SDK for JavaScript
AWS SDK for JavaScript v2	2.68.0 以降にアップグレード。	あり	AWS SDK for JavaScript

この SDK または製品を使用する場合	この SDK バージョンにアップグレード	Sigv4 を使用するために、クライアントでコード変更が必要ですか。	SDK ドキュメントへのリンク
AWS SDK for JavaScript v3	現在のところ、何もする必要はありません。主要バージョン V3 in Q3 2019 にアップグレード。	なし	AWS SDK for JavaScript
AWS SDK for PHP v1	最新バージョンの PHP、または少なくとも S3 クライアントの設定で署名パラメータを v4 に設定した v2.7.4 にアップグレードすることをお勧めします。	あり	AWS SDK for PHP

この SDK または製品を使用する場合	この SDK バージョンにアップグレード	Sigv4 を使用するために、クライアントでコード変更が必要ですか。	SDK ドキュメントへのリンク
AWS SDK for PHP v2	最新バージョンの PHP、または少なくとも S3 クライアントの設定で署名パラメータを v4 に設定した v2.7.4 にアップグレードすることをお勧めします。	なし	AWS SDK for PHP
AWS SDK for PHP v3	SDK のアップグレードは不要です。	なし	AWS SDK for PHP
Boto2	Boto2 v2.49.0 にアップグレード。	あり	Boto 2 アップグレード
Boto3	1.5.71 (Botocore)、1.4.6 (Boto3) にアップグレード。	あり	Boto 3 - AWS SDK for Python

この SDK または製品を使用する場合	この SDK バージョンにアップグレード	Sigv4 を使用するために、クライアントでコード変更が必要ですか。	SDK ドキュメントへのリンク
AWS CLI	1.11.108 にアップグレード。	あり	AWS Command Line Interface
AWS CLI v2 (プレビュー)	SDK のアップグレードは不要です。	なし	AWS Command Line Interface バージョン 2
AWS SDK for Ruby v1	Ruby V3 にアップグレード。	あり	Ruby V3 for AWS
AWS SDK for Ruby v2	Ruby V3 にアップグレード。	あり	Ruby V3 for AWS
AWS SDK for Ruby v3	SDK のアップグレードは不要です。	なし	Ruby V3 for AWS
Go	SDK のアップグレードは不要です。	なし	AWS SDK for Go
C++	SDK のアップグレードは不要です。	なし	AWS SDK for C++

AWS Tools for Windows PowerShell または AWS Tools for PowerShell Core

3.3.0.0 より前のモジュールバージョンを使用している場合は、3.3.0.0 にアップグレードする必要があります。

バージョン情報を取得するには、Get-Module コマンドレットを使用します。

```
Get-Module –Name AWSPowershell
Get-Module –Name AWSPowershell.NetCore
```

3.3.0.0 バージョンに更新するには、Update-Module コマンドレットを使用します。

```
Update-Module –Name AWSPowershell
Update-Module –Name AWSPowershell.NetCore
```

署名バージョン 2 をトラフィックで送信する、7 日間以上で有効な署名付き URL を使用できます。

REST API を使用した Amazon S3 での開発

Amazon S3 は、プログラミング言語に依存しないアーキテクチャとして設計されており、サポートされているインターフェイスを使用してオブジェクトを保存し、取得します。

Amazon S3 は現在、REST インターフェイスを提供しています。REST を使うと、メタデータは HTTP ヘッダーで返されます。4 KB (本文を含まない) 以下の HTTP リクエストしかサポートされないため、使用できるメタデータの量が制限されます。REST API は、Amazon S3 に対する HTTP インターフェイスです。REST を使用すると、標準 HTTP リクエストを使用してバケットとオブジェクトを作成、取得、および削除できます。

REST API を使用する場合、HTTP をサポートする任意のツールキットを使用できます。匿名で読み取り可能なオブジェクトであれば、ブラウザを使用して取得することもできます。

REST API は標準の HTTP ヘッダーとステータスコードを使用するため、標準のブラウザとツールキットが予期したとおりに機能します。一部のエリアでは、HTTP に機能が追加されています (たとえば、アクセスコントロールをサポートするヘッダーを追加しました)。このように新機能を追加する場合、できるだけ標準 HTTP 書式の使用法に合致するように最善を尽くしました。

REST API を使用したリクエスト送信の詳細については、「REST API を使用したリクエストの実行」を参照してください。REST API を使用する際に留意すべき考慮事項については、以下のトピックを参照してください。

Amazon S3 REST API の詳細については、「Amazon Simple Storage Service API リファレンス」を参照してください。

トピック

- リクエストルーティング

リクエストルーティング

CreateBucketConfiguration を含む CreateBucket API を使用して作成されたバケットに対してリクエストを実行するプログラムは、リダイレクトをサポートしている必要があります。また、DNS TTL に従わない一部のクライアントで問題が発生する場合があります。

このセクションでは、Amazon S3 で使用するサービスまたはアプリケーションを設計するときに考慮すべきルーティングおよび DNS の問題について説明します。

リクエストのリダイレクトと REST API

Amazon S3 はドメインネームシステム (DNS) を使用して、リクエストを処理できる施設にルーティングします。このシステムは有効に機能しますが、一時的なルーティングエラーが発生する場合があります。リクエストが誤った Amazon S3 ロケーションに到達した場合、Amazon S3 は、リクエストを新しいエンドポイントに再送するようリクエスタに指示する一時的なリダイレクトで応答します。リクエストの形式が正しくない場合、Amazon S3 は恒久的なリダイレクトを使用して、リクエストを正しく実行する方法について指示します。

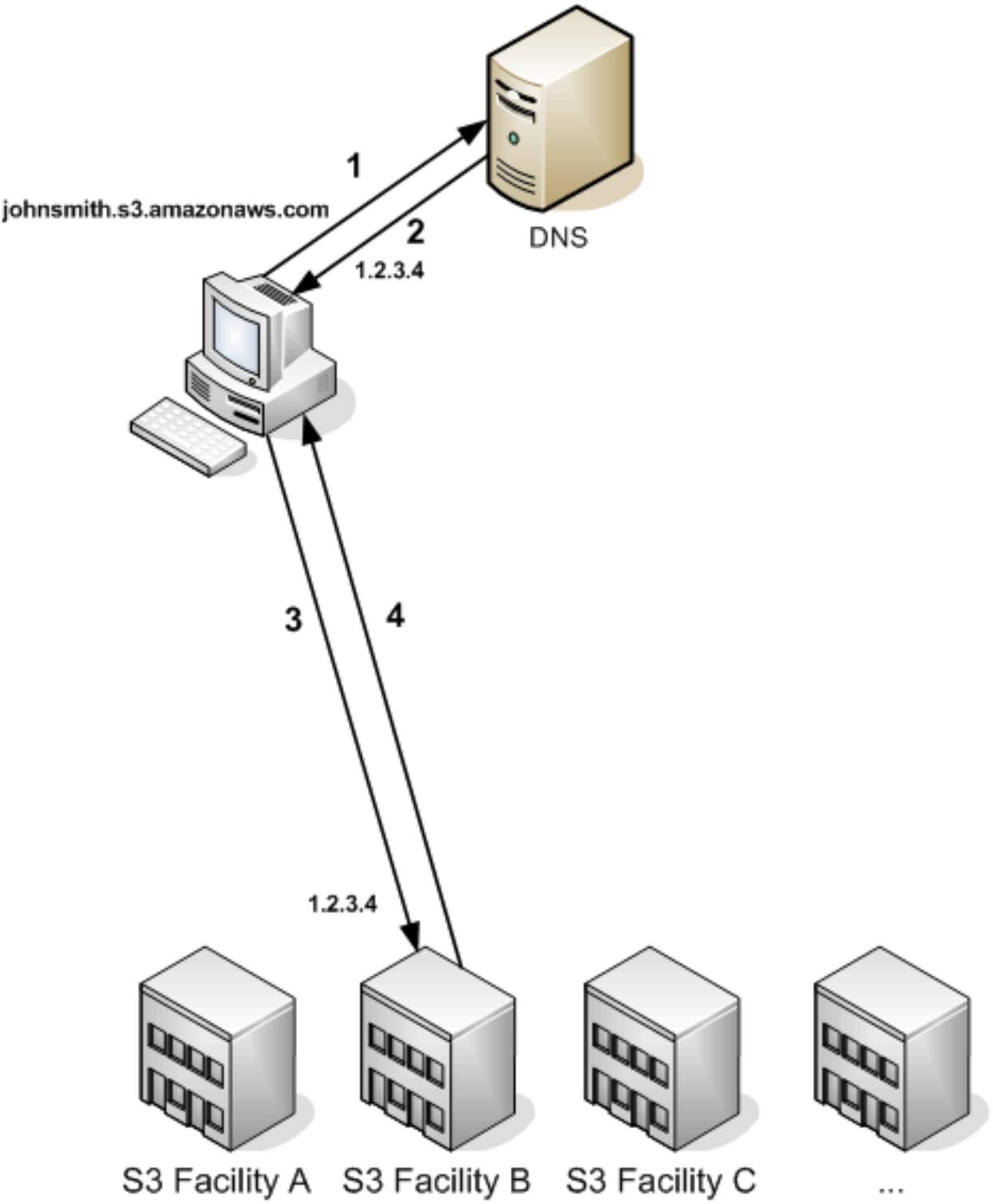
> **Important**
>
> この機能を使用するには、Amazon S3 リダイレクトレスポンスを処理できるアプリケーションが必要です。唯一の例外は、`<CreateBucketConfiguration>` なしで作成されたバケットのみを使用するアプリケーションです。ロケーションの制約の詳細については、「Amazon S3 バケットに対するアクセスと一覧表示」を参照してください。
> 2019 年 3 月 20 日より後に開始されたすべてのリージョンで、リクエストが間違った Amazon S3 ロケーションに到着した場合、Amazon S3 は HTTP 400 Bad Request エラーを返します。

AWS リージョン の有効化または無効化の詳細については、「AWS 全般のリファレンス」の「AWS リージョン とエンドポイント」を参照してください。

トピック

DNS ルーティング

DNS ルーティングは、リクエストを適切な Amazon S3 施設にルーティングします。以下の図とステップは、DNS ルーティングの例を示しています。



DNS ルーティングリクエストのステップ

1. クライアントが、Amazon S3 に保存されているオブジェクトを取得するために DNS リクエストを行います。

2. クライアントが、リクエストを処理できる施設の 1 つまたは複数の IP アドレスを受け取ります。この例で、IP アドレスは施設 B のものです。
3. クライアントが、Amazon S3 施設 B へのリクエストを行います。
4. 施設 B が、オブジェクトのコピーをクライアントに返します。

一時的なリクエストのリダイレクト

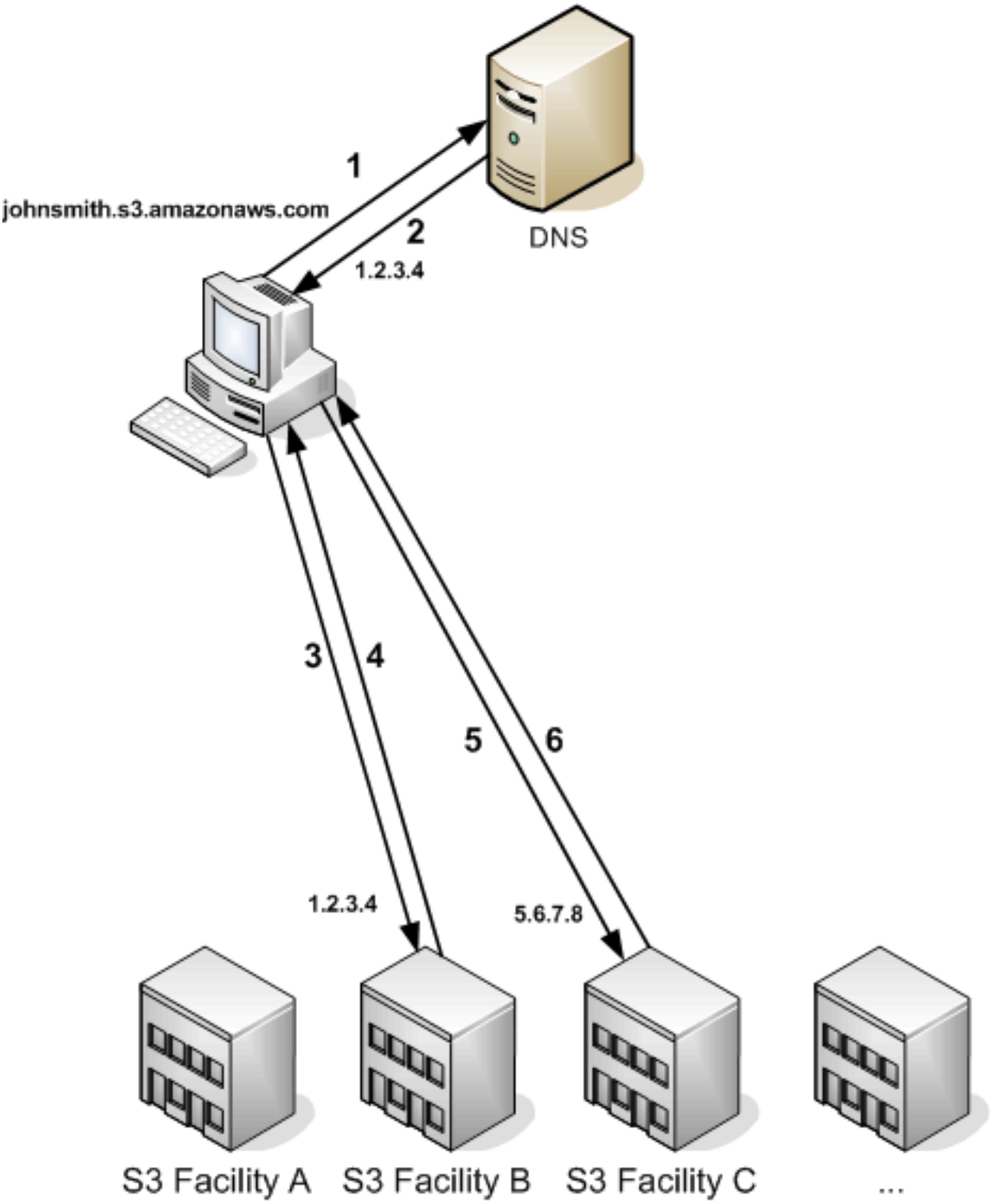
一時的なリダイレクトとは、リクエストを異なるエンドポイントに再送する必要があることをリクエスタに知らせる一種のエラーレスポンスです。Amazon S3 が持つ分散型の特質により、リクエストが誤った施設に一時的にルーティングされる場合があります。このエラーは、大半の場合にバケットが作成または削除された直後に発生します。

たとえば、新しいバケットを作成し、ただちにバケットへのリクエストを実行する場合、バケットのロケーションの制約によっては一時的にリダイレクトされる可能性もあります。AWS リージョン の米国東部 (バージニア北部) にバケットを作成した場合、これはデフォルトの Amazon S3 エンドポイントでもあるため、そのバケットへのリクエストはリダイレクトされることはありません。

一方、その他のリージョンにバケットを作成した場合、バケットの DNS エントリは伝播されますが、バケットに対するリクエストはデフォルトのエンドポイントに転送されます。デフォルトのエンドポイントは、リクエストを正しいエンドポイントにリダイレクトし、HTTP 302 レスポンスを返します。一時的なリダイレクトには正しい施設への URI が含まれており、これを使用して直ちにリクエストを再送できます。

> **Important**
>
> 以前のリダイレクトレスポンスで返されたエンドポイントを再利用しないでください。このようなエンドポイントは (長期的にも) 機能するように見えますが、予期しない結果を招くことがあり、最終的には通知なしに失敗します。

以下の図とステップは、一時的なリダイレクトの例を示しています。



一時的なリクエストのリダイレクトのステップ

1. クライアントが、Amazon S3 に保存されているオブジェクトを取得するために DNS リクエストを行います。

2. クライアントが、リクエストを処理できる施設の 1 つまたは複数の IP アドレスを受け取ります。
3. クライアントが、Amazon S3 施設 B へのリクエストを行います。
4. 施設 B がリダイレクトを返して、オブジェクトがロケーション C から利用可能であることを示します。
5. クライアントがリクエストを施設 C に再送します。
6. 施設 C がオブジェクトのコピーを返します。

永続的なリクエストのリダイレクト

恒久的なリダイレクトは、リクエストがリソースのアドレス指定を不適切に行ったことを示します。たとえば、パス形式のリクエストを使用して、<CreateBucketConfiguration> によって作成されたバケットにアクセスした場合、恒久的なリダイレクトが発生します。詳細については、「Amazon S3 バケットに対するアクセスと一覧表示」を参照してください。

開発中にこれらのエラーを見つけやすいように、このタイプのリダイレクトには、リクエストを正しいロケーションに自動的に送信するためのロケーション HTTP ヘッダーが含まれていません。正しい Amazon S3 エンドポイントを使用するために、生成される XML エラードキュメントを参照してください。

リクエストのリダイレクトの例

一時的なリクエストのリダイレクトレスポンスの例を以下に示します。

REST API の一時的なリダイレクトレスポンス

```
HTTP/1.1 307 Temporary Redirect
Location: http://awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com/photos/puppy.jpg?
rk=e2c69a31
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Fri, 12 Oct 2007 01:12:56 GMT
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the specified temporary endpoint.
  Continue to use the original request endpoint for future requests.</Message>
```

```
  <Endpoint>awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com</Endpoint>
</Error>
```

SOAP API の一時的なリダイレクトレスポンス

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.TemporaryRedirect</Faultcode>
    <Faultstring>Please re-send this request to the specified temporary endpoint.
    Continue to use the original request endpoint for future requests.</Faultstring>
    <Detail>
      <Bucket>images</Bucket>
      <Endpoint>s3-gztb4pa9sq.amazonaws.com</Endpoint>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

DNS に関する考慮事項

Amazon S3 の設計要件の 1 つは、非常に高い可用性です。この要件を満たす方法の 1 つとして、DNS 内の Amazon S3 エンドポイントに関連付られた IP アドレスを必要に応じて更新します。これらの変更は、存続時間の短いクライアントには自動的に反映されますが、存続時間の長いクライアントには反映されない場合があります。存続時間の長いクライアントの場合、これらの変更を活用するには、Amazon S3 エンドポイントを定期的に再解決するための特殊なアクションを実行する必要があります。仮想マシン (VM) の詳細については、次のトピックを参照してください。

- Java の場合、Sun の JVM はデフォルトで DNS ルックアップを永続的にキャッシュします。この動作を変更する方法については、InetAddress のドキュメントの「InetAddress キャッシング」セクションを参照してください。
- PHP の場合、ほとんどの一般的な導入構成で実行される永続的な PHP VM は、VM が再起動されるまで DNS ルックアップをキャッシュに保持します。getHostByName についての PHP ドキュメントを参照してください。

REST エラーと SOAP エラーの処理

トピック

このセクションでは、REST エラーと SOAP エラー、およびその処理方法について説明します。

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

REST エラーレスポンス

REST リクエストがエラーになった場合、HTTP の応答には次のものが含まれます。

- レスポンス本文としての XML エラードキュメント
- コンテンツタイプ: application/xml
- 該当する HTTP ステータスコード (3xx、4xx、または 5xx)

REST エラーレスポンスの例を次に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/mybucket/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

Amazon S3 エラーの詳細については、「ErrorCodeList」を参照してください。

レスポンスヘッダー

すべてのオペレーションで返されるレスポンスヘッダーを次に示します。

- `x-amz-request-id`: システムによって各リクエストに割り当てられた一意の ID。万一 Amazon S3 の使用時に問題が発生した場合でも、Amazon はこの ID を使用して問題のトラブルシューティングを行うことができます。
- `x-amz-id-2`:トラブルシューティングに役立つ特殊なトークン。

エラーレスポンス

Amazon S3 リクエストがエラーになると、クライアントはエラーレスポンスを受け取ります。エラーレスポンスの正確な形式は API 固有です。たとえば、REST エラーレスポンスは SOAP エラーレスポンスとは異なります。ただし、すべてのエラーレスポンスには共通の要素があります。

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

エラーコード

エラーコードは、エラー状態を個別に識別する文字列です。エラーを検出してタイプ別に処理するプログラムによって読み取りおよび解釈されるためのものです。多くのエラーコードは SOAP と REST API で共通ですが、API 固有のものもあります。たとえば、NoSuchKey は共通ですが、UnexpectedContent は、無効な REST リクエストへのレスポンスでのみ発生します。SOAP フォルトコードには、どの場合でも、エラーコードの表に示すプレフィックスが付いているので、NoSuchKey エラーは SOAP では実際には Client.NoSuchKey として返されます。

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

エラーメッセージ

エラーメッセージには、エラー状態の一般的な説明が英語で含まれます。これは人が理解できるようにするためのものです。シンプルなプログラムの場合、処理できない、または処理されないエラー状態が発生すると、メッセージがエンドユーザーに直接表示されます。より徹底したエラー処理を備え、適切に国際化されている洗練されたプログラムでは、エラーメッセージが無視される傾向にあります。

詳細情報

多くのエラーレスポンスには、プログラミングエラーを診断する開発者が読んで理解できるように追加の構造化データが含まれています。たとえば、REST PUT リクエストと共に Content-MD5 ヘッダーを送信し、これがサーバーで計算されたダイジェストと一致しない場合、BadDigest エラーが発生します。エラーレスポンスには、計算されたダイジェスト、指定されたダイジェストなど、詳細要素も含まれます。開発中にこの情報を使用してエラーを診断できます。実稼働では、正常に動作するプログラムによりこの情報がエラーログに記録されることがあります。

SOAP エラーレスポンス

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

SOAP の場合、エラー結果は SOAP フォルトとして HTTP レスポンスコード 500 と共にクライアントに返されます。SOAP フォルトを受け取っていない場合、リクエストは成功しています。Amazon S3 SOAP フォルトコードは、標準の SOAP 1.1 フォルトコード (「Server」または「Client」) に Amazon S3 固有のエラーコードが連結した構成になっています。たとえば、「Server.InternalError」や「Client.NoSuchBucket」などです。SOAP フォルト文字列要素には、人が読むことができる英語の汎用エラーメッセージが含まれます。また、SOAP フォルト詳細要素には、エラーに関連するさまざまな情報が含まれます。

たとえば、「Fred」というオブジェクトを削除しようとして、このオブジェクトが存在しなかった場合、SOAP レスポンスの本文には「NoSuchKey」という SOAP エラーが含まれます。

Example

```
<soapenv:Body>
```

```
  <soapenv:Fault>
    <Faultcode>soapenv:Client.NoSuchKey</Faultcode>
    <Faultstring>The specified key does not exist.</Faultstring>
    <Detail>
      <Key>Fred</Key>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

Amazon S3 エラーの詳細については、「ErrorCodeList」を参照してください。

Amazon S3 のエラーに関するベストプラクティス

Amazon S3 と連携するアプリケーションを設計するときは、Amazon S3 のエラーを適切に処理することが重要です。このセクションでは、アプリケーションの設計時に考慮すべき問題について説明します。

InternalError の場合は再試行する

内部エラーとは、Amazon S3 内で発生するエラーです。

InternalError レスポンスを受け取った場合、リクエストが処理されていない可能性があります。たとえば、PUT リクエストが InternalError を返す場合、それ以降の GET は古い値を取得することもあれば、更新された値を取得することもあります。

Amazon S3 が InternalError レスポンスを返す場合は、リクエストを再試行してください。

SlowDown エラーを繰り返すアプリケーションを調整する

S3 は分散システムの常として、意図的であるかどうかに関係なくリソースの過剰な消費を検出して対応する保護メカニズムを備えています。SlowDown エラーは、リクエストレートが高いためにこのようなメカニズムのいずれかが引き起こされた場合に発生することがあります。リクエストレートを低くすると、この種のエラーは減るかなくなります。一般に、ほとんどのユーザーはこのようなエラーを日常的に経験することはありません。ただし、詳細を知りたい場合や、SlowDown エラーが頻繁にまたは予期せず発生する場合は、Amazon S3 デベロッパーフォーラムに投稿するか、または AWS Support (https://aws.amazon.com/premiumsupport/) にサインアップしてください。

エラーを分離する

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

Amazon S3 には、SOAP と REST API の両方で使用されるエラーコードが用意されています。SOAP API は標準の Amazon S3 エラーコードを返します。REST API は標準の HTTP サーバーのように見え、既存の HTTP クライアント (例: ブラウザ、HTTP クライアントライブラリ、プロキシ、キャッシュなど) を操作するように設計されています。HTTP クライアントがエラーを適切に処理できるように、各 Amazon S3 エラーは HTTP ステータスコードにマッピングされています。

HTTP ステータスコードは Amazon S3 エラーコードよりも内容がおおまかであるため、エラーに関する情報は少なくなります。たとえば、`NoSuchKey` と `NoSuchBucket` という Amazon S3 エラーはどちらも `HTTP 404 Not Found` ステータスコードにマッピングされます。

HTTP ステータスコードに含まれるエラー情報は少なくなりますが、クライアントが HTTP を認識できて Amazon S3 API を認識できない場合も、通常はエラーが適切に処理されます。

このため、エラーを処理するときや Amazon S3 エラーをエンドユーザーに報告するときは、HTTP ステータスコードではなく、エラーに関する情報が多く含まれる Amazon S3 エラーコードを使用します。さらに、アプリケーションをデバッグするときは、人が読み取り可能な XML エラーレスポンスの <Details> 要素も参照してください。

デベロッパーリファレンス

この付録には以下のセクションがあります。

トピック

付録 A: SOAP API の使用

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

このセクションでは、Amazon S3 の SOAP API に固有の情報について説明します。

> **Note**
>
> SOAP リクエストは、認証済みの場合も匿名の場合も、SSL を使用して Amazon S3 に送信する必要があります。HTTP を介して SOAP リクエストが送信された場合、Amazon S3 はエラーを返します。

トピック

一般的な SOAP API 要素

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

HTTP で SOAP 1.1 を使用して、Amazon S3 と通信することができます。Amazon S3 の API をマシンで読み取り可能な方法で記述している Amazon S3 の WSDL は、https://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl から入手できます。Amazon S3 のスキーマは、https://doc.s3.amazonaws.com/2006-03-01/AmazonS3.xsd から入手できます。

ほとんどのユーザーは、言語および開発環境に合わせてカスタマイズされた SOAP ツールキットを使用して Amazon S3 と通信します。ツールキットごとに、Amazon S3 の API を公開する方法は異なります。使用方法については、特定のツールキットのドキュメントを参照してください。このセクションでは、「オンライン」上での XML リクエストと XML レスポンスを示すことにより、ツールキットに依存しない Amazon S3 の SOAP オペレーションについて説明します。

一般的な要素

いずれの SOAP リクエストにも、認証に関連する次の要素を含めることができます。

- `AWSAccessKeyId`: リクエスタの AWS アクセスキー ID
- `Timestamp`: システムの現在の時刻
- `Signature`: リクエストの署名

SOAP リクエストの認証方法

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

リクエスト元となるプリンシパルのアイデンティティを確立するには、すべての非匿名リクエストは認証情報を含んでいる必要があります。SOAP では、認証情報は SOAP リクエストの以下の要素に置かれています。

- お客様の AWS アクセスキー ID。

  > **Note**
  >
  > 認証済みの SOAP リクエストを行う場合、一時セキュリティ証明書はサポートされません。証明書の種類の詳細については、リクエストの実行 を参照してください。

- `Timestamp`: これは、`2009-01-01T12:00:00.000Z` など、協定世界時 (グリニッジ標準時) タイムゾーンの dateTime (http://www.w3.org/TR/xmlschema-2/#dateTime を参照) である必要があります。このタイムスタンプが Amazon S3 サーバーの時刻から 15 分を過ぎている場合、認可は失敗します。

- `Signature`: AWS のシークレットアクセスキーをキーとして使用し、AmazonS3 + オペレーション + タイムスタンプの連結文字で生成される RFC 2104 の HMAC-SHA1 ダイジェスト (http://www.ietf.org/rfc/rfc2104.txt を参照)。たとえば、以下の CreateBucket サンプルリクエストでは、署名要素に「AmazonS3CreateBucket2009-01-01T12:00:00.000Z」という値の HMAC-SHA1 ダイジェストが含まれています。

たとえば、以下の CreateBucket サンプルリクエストでは、署名要素に「AmazonS3CreateBucket2009-01-01T12:00:00.000Z」という値の HMAC-SHA1 ダイジェストが含まれています。

Example

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Acl>private</Acl>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-01-01T12:00:00.000Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

> **Note**
>
> SOAP リクエストは、認証済みの場合も匿名の場合も、SSL を使用して Amazon S3 に送信する必要があります。HTTP を介して SOAP リクエストが送信された場合、Amazon S3 はエラーを返します。

> **Important**
>
> 時間の精度の桁数をどこで切り捨てるかにはさまざまな解釈があるため、.NET ユーザーは過度に詳細なタイムスタンプを Amazon S3 に送信しないようようにする必要があります。ミリ秒での精度で `DateTime` オブジェクトを手動で構築することによって、これを達成することができます。

SOAP のアクセスポリシーの設定

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

アクセスコントロールは、バケットまたはオブジェクトの記述時に、「AccessControlList」要素と一緒に `CreateBucket`、`PutObjectInline`、または `PutObject` へのリクエストを含めることによって設定できます。AccessControlList 要素については Amazon S3 用 Identity and Access Management で説明しています。これらのオペレーションでアクセスコントロールリストが指定されていない場合、リクエスタに FULL_CONTROL アクセスを付与するデフォルトのアクセスポリシーでリソースを生成します (これは該当のリクエストが、既に存在するオブジェクトへの PutObjectInline または PutObject リクエストである場合にも当てはまります)。

以下に示すリクエストは、オブジェクトにデータを書き込み、匿名のプリンシパルによるオブジェクトの読み取りを可能にし、特定のユーザーにバケットへの FULL_CONTROL の権限を付与します (ほとんどの開発者は、自分で自分のバケットに FULL_CONTROL アクセスを付与したいと考えるでしょう)。

Example

以下に示すリクエストは、オブジェクトにデータを書き込み、匿名のプリンシパルによるオブジェクトの読み取りを可能にします。

`Sample Request`

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
```

```
      <Grantee xsi:type="CanonicalUser">
        <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>&quot828ef3fdfa96f00ad9f27c383fc9ac7f&quot</ETag>
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

既存のバケットまたはオブジェクトのアクセスコントロールポリシーは、GetBucketAccessControlPolicy、GetObjectAccessControlPolicy、SetBucketAccessCon メソッドを使って読み込みまたは設定できます。詳細については、これらのメソッドの詳しい説明を参照してください。

付録 b: リクエストの認証 (AWS 署名バージョン 2)

> **Important**
>
> このセクションでは AWS 署名バージョン 2 を使用してリクエストを認証する方法を説明します。署名バージョン 2 は無効 (非推奨) にされているため、Amazon S3 は署名バージョン 4 を使用して署名された API リクエストのみを受け入れます。詳細については、「AWSAmazon S3 の 署名バージョン 2 がオフになっている (非推奨)」を参照してください。

すべての AWS リージョン は署名バージョン 4 をサポートし、新しいリージョンでサポートされるのはこのバージョンのみです。詳細については、Amazon Simple Storage Service API リファレンスの「リクエストの認証 (AWS 署名バージョン 4)」を参照してください。Amazon S3 では、リクエストの署名にどの API 署名バージョンが使用されているかを識別できます。ビジネスへの影響を防ぐために、いずれかのワークフローが署名バージョン 2 の署名を利用しているかどうかを識別し、それらを署名バージョン 4 を使用するようにアップグレードすることが重要です。

- CloudTrail のイベントログ (推奨オプション) を使用している場合にそのようなリクエストをクエリおよび識別する方法については、「CloudTrail を使用した Amazon S3 Signature Version 2 リクエストの識別」を参照してください。
- Amazon S3 のサーバーアクセスログを使用している場合は、「Amazon S3 アクセスログを使用した Signature Version 2 リクエストの識別」を参照してください。

トピック

- REST API を使用したリクエストの認証
- REST リクエストの署名と認証
- POST (AWS 署名バージョン 2) を使用したブラウザベースのアップロード

REST API を使用したリクエストの認証

REST を使用して Amazon S3 にアクセスする場合は、リクエストが認証されるように、リクエストに以下の項目を指定する必要があります。

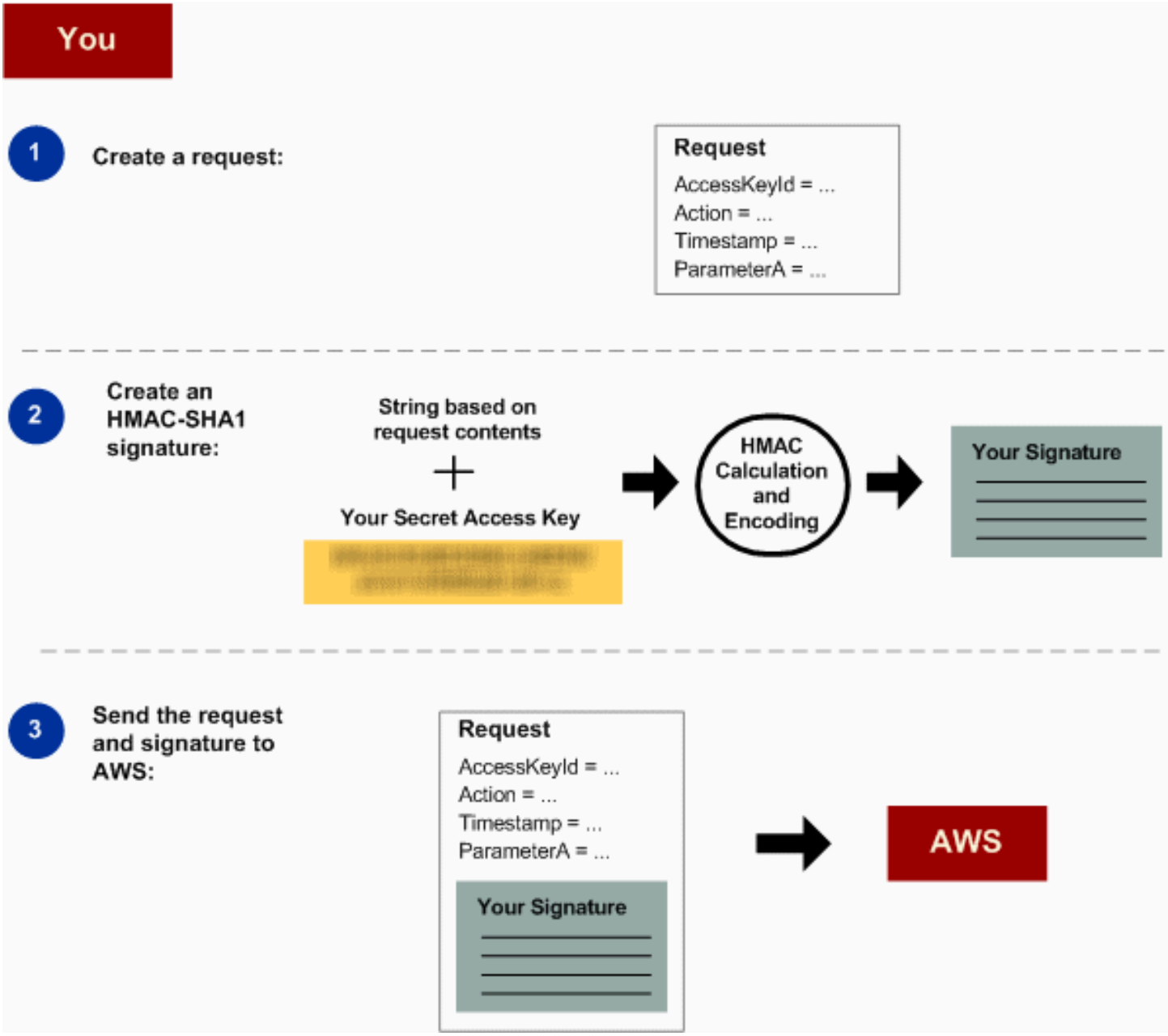
リクエストの要素

- AWS アクセスキー ID – 各リクエストには、リクエストの送信に使用するアイデンティティのアクセスキー ID を含める必要があります。
- 署名 – 各リクエストには、リクエストの有効な署名を含める必要があります。含まれていない場合、リクエストは拒否されます。

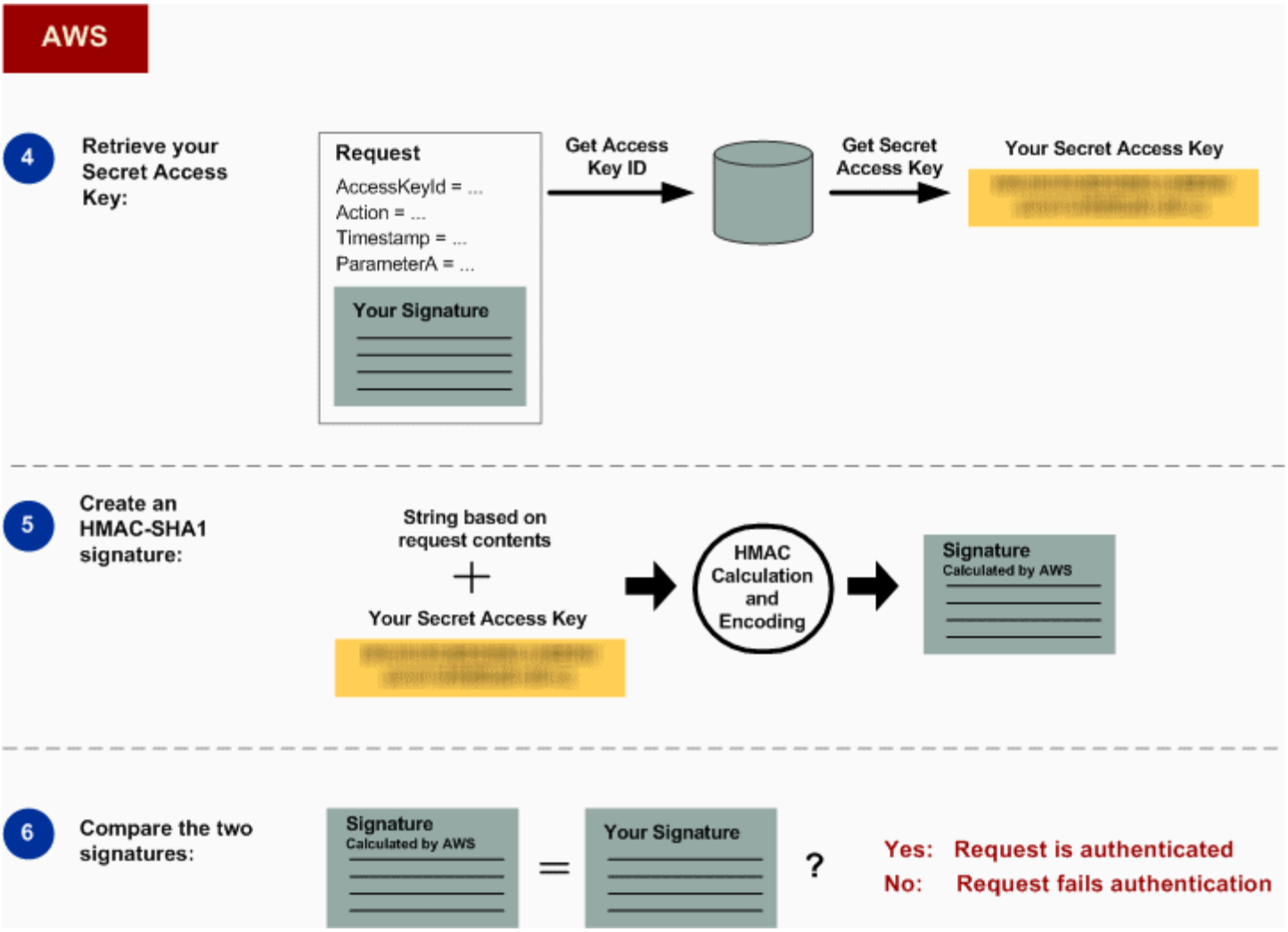
  リクエスト署名は、シークレットアクセスキーを使用して計算されます。シークレットアクセスキーは、自分と AWS のみが知っている共有の秘密です。
- タイムスタンプ – 各リクエストには、リクエストを作成した日時 (UTC) を文字列として含める必要があります。
- 日付 – 各リクエストには、リクエストのタイムスタンプを含める必要があります。

  使用する API アクションによっては、タイムスタンプを追加するかわりに、リクエストの有効期限を示す日時を指定できます。特定のアクションに必要な項目を確認するには、そのアクションの認証のトピックを参照してください。

Amazon S3 へのリクエストを認証する一般的なステップを以下に示します。この説明では、必要なセキュリティ認証情報、アクセスキー ID、およびシークレットアクセスキーをお客様が持っていることを前提としています。



1	AWSにリクエストします。
2	シークレットアクセスキーを使用して署名を計算します。
3	Amazon S3 にリクエストを送信します。リクエストにアクセスキー ID と署名を含めます。次 の 3 つのステップは、Amazon S3 で実行されます。



4	Amazon S3 がアクセスキー ID を使用してシークレットアクセスキーを調べます。
5	リクエストで送信した署名を計算するために使用したものと同じアルゴリズムを使用して、Amazon S3 がリクエストのデータとシークレットアクセスキーから署名を計算します。
6	Amazon S3 が生成した署名がリクエストで送信したものと一致した場合、リクエストは本物とみなされます。署名が一致しなかった場合、リクエストは破棄され、Amazon S3 はエラーレスポンスを返します。

詳細な認証情報

REST 認証の詳細については、「REST リクエストの署名と認証」を参照してください。

REST リクエストの署名と認証

トピック

> **Note**
>
> このトピックでは、署名バージョン 2 を使用してリクエストを認証する方法を説明します。Amazon S3 では、現在、最新の署名バージョン 4 がサポートされています。この最新の署名バージョンはすべてのリージョンでサポートされており、2014 年 1 月 30 日以降は、新しいリージョンではバージョン 4 の署名のみがサポートされます。詳細については、Amazon Simple Storage Service API リファレンスの「リクエストの認証 (AWS 署名バージョン 4)」を参照してください。

認証とは、自身のアイデンティティをシステムに証明するプロセスのことをいいます。アイデンティティは、Amazon S3 のアクセス管理の判断における重要な要因です。リクエストの許可または拒否は、リクエスタのアイデンティティに部分的に基づいています。たとえば、バケットを作成する権利は、登録開発者用に予約されています。また、バケットにオブジェクトを作成する権利は、対象のバケット所有者用に (デフォルトで) 予約されています。開発者としてこれらの権限を実行するリクエストを行うので、そのリクエストを認証することで、自身のアイデンティティをシステムに対して証明する必要があります。このセクションでは、その方法を説明します。

> **Note**
>
> このセクションの内容は HTTP POST には適用されません。詳細については、「POST (AWS 署名バージョン 2) を使用したブラウザベースのアップロード」を参照してください。

Amazon S3 の REST API では、キーを使用して生成される HMAC (ハッシュメッセージ認証コード) に基づくカスタムの HTTP スキーマが認証に使用されます。リクエストを認証するには、まず、選択されているリクエストの要素を連結し、文字列を作成します。次に、AWS シークレットアクセスキーを使用して、その文字列の HMAC を計算します。正式な呼び名ではありませんが、このプロセスのことを「リクエストへの署名」と呼び、HMAC アルゴリズムの出力を署名と呼びます。これは、このプロセスが実際の署名のセキュリティプロパティを真似ているからです。最後に、このセクションで説明する構文を使用して、この署名をリクエストのパラメータとして追加します。

システムは、認証済みリクエストを受け取るとき、リクエスト送信者が所有している AWS シークレットアクセスキーを取得し、同様の方法でそのアクセスキーを使用して、受信したメッセージの署名を計算します。そして、計算した署名と、リクエスタから提示された署名を比較します。2 つの署名が一致したら、リクエスタは AWS シークレットアクセスキーにアクセスできると判断されるため、システムはそのキーの発行先となるプリンシパルの権限をサポートします。2 つの署名が一致しない場合、リクエストは中断し、エラーメッセージが返されます。

Example 認証された Amazon S3 の REST リクエスト

```
GET /photos/puppy.jpg HTTP/1.1
Host: awsexamplebucket1.us-west-1.s3.amazonaws.com
Date: Tue, 27 Mar 2007 19:36:42 +0000

Authorization: AWS AKIAIOSFODNN7EXAMPLE:
qgk2+6Sv9/oM7G3qLEjTH1a1l1g=
```

一時的なセキュリティ認証情報の使用

一時的なセキュリティ認証情報を使用してリクエストに署名する場合は (「リクエストの実行」を参照)、`x-amz-security-token` ヘッダーを追加して、対応するセキュリティトークンをリクエストに含める必要があります。

AWS Security Token Service API を使用して一時的なセキュリティ認証情報を取得すると、そのレスポンスには、一時的なセキュリティ認証情報とセッショントークンが含まれます。リクエストを Amazon S3 に送信するときに、`x-amz-security-token` ヘッダーでセッショントークン値を

指定します。IAM が提供する AWS Security Token Service API については、AWS Security Token Service API リファレンスガイドの「アクション」を参照してください。

認証ヘッダー

Amazon S3 の REST API は、標準の HTTP ヘッダーの Authorization を使用して認証情報を渡します。(標準ヘッダーの名前とは異なり、ヘッダーに含まれるのは承認ではなく認証情報です。) Amazon S3 の認証スキームにおける Authorization ヘッダーの形式は次のとおりです。

```
Authorization: AWS AWSAccessKeyId:Signature
```

デベロッパーには、登録時に AWS アクセスキー ID と AWS シークレットアクセスキーが発行されます。リクエストを認証するために、AWSAccessKeyId 要素は、署名の計算に使用されたアクセスキー ID を識別するほか、リクエストを行っている開発者も間接的に識別します。

Signature 要素は、リクエストから選択した要素の RFC 2104HMAC-SHA1 です。したがって、Authorization ヘッダーの Signature 部分はリクエストによって異なります。システムによって計算されたリクエストの署名が、リクエストに含まれる Signature と一致する場合は、リクエスタが AWS シークレットアクセスキーを所有していることになります。その後、リクエストは、キーの発行対象者である開発者のアイデンティティと権限に従って処理されます。

以下は擬似文法で、Authorization リクエストヘッダーの構文例を示しています。(例中の \n は Unicode のコードポイント U+000A を意味しています。これは通常改行と呼ばれています)。

```
Authorization = "AWS" + " " + AWSAccessKeyId + ":" + Signature;

Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of(YourSecretAccessKey), UTF-8-Encoding-
Of( StringToSign ) ) );

StringToSign = HTTP-Verb + "\n" +
 Content-MD5 + "\n" +
 Content-Type + "\n" +
 Date + "\n" +
 CanonicalizedAmzHeaders +
 CanonicalizedResource;

CanonicalizedResource = [ "/" + Bucket ] +
 <HTTP-Request-URI, from the protocol name up to the query string> +
 [ subresource, if present. For example "?acl", "?location", or "?logging"];
```

```
CanonicalizedAmzHeaders = <described below>
```

HMAC-SHA1 は、RFC 2104 の Keyed-Hashing for Message Authentication で定義されているアルゴリズムです。このアルゴリズムは、キーとメッセージの 2 つのバイト文字列を入力として取ります。Amazon S3 のリクエストの認証では、AWS のシークレットアクセスキー (YourSecretAccessKey) をキーとして使用し、StringToSign の UTF-8 エンコーディングをメッセージとして使用します。また、HMAC-SHA1 の出力もバイト文字列で、これはダイジェストと呼ばれます。Signature リクエストパラメータは、このダイジェストをエンコードする Base64 によって作成されます。

署名のためのリクエストの標準化

既に説明したように、システムは認証済みリクエストを受け取るときに、計算されたリクエスト署名と StringToSign のリクエストで指定された署名を比較します。そのため、署名は Amazon S3 で使用されているものと同じ方法で計算する必要があります。署名のためにリクエストを承認済み形式にするプロセスは正規化と言います。

CanonicalizedResource 要素の作成

CanonicalizedResource は、リクエストの対象となる Amazon S3 のリソースを表します。REST リクエストのこのリソースは次のように作成します。

プロセスを起動する

1	空の文字列 ("") で開始します。
2	HTTP ホストヘッダー (仮想ホスト形式) を使用してバケットが指定されているリクエストについては、バケット名の前に "/" を付けます (例: 「/bucketname」)。パス形式のリクエスト、およびバケットを処理しないリクエストの場合は、何も行いません。仮想ホスト形式のリクエストの詳細については、「バケットの仮想ホスティング」を参照してください。 仮想ホスティング形式のリクエスト「https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg」の場合、CanonicalizedResource は「/awsexamplebucket1」です。 パス形式のリクエスト「https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg」の場合、CanonicalizedResource は "" です。
3	デコードされていない HTTP リクエスト URI のパス部分を追加します (クエリ文字列まで、ただし、その文字列は含みません)。

	仮想ホスト形式のリクエスト「https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg」の場合、CanonicalizedResource は「/awsexamplebucket1/photos/puppy.jpg」です。 パス形式のリクエスト「https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg」の場合、CanonicalizedResource は「/awsexamplebucket1/photos/puppy.jpg」です。この時点で、CanonicalizedResource は、仮想ホスト形式とパス形式の両方のリクエストで同じです。 GET サービスなどのバケットを指定しないリクエストの場合は、「/」を追加します。
4	?versioning 、?location 、?acl、?lifecycle 、?versionid などのサブリソースを処理するリクエストの場合は、サブリソース、そのサブリソースの値 (存在する場合)、および疑問符を追加します。複数のサブリソースは、名前のアルファベット順に並べ替えて "&" で区切る必要があります。たとえば、?acl&versionId=*value* のようにする必要があります。 CanonicalizedResource 要素を作成するときに含める必要があるサブリソースは、acl、lifecycle、location、logging、notification、partNumber、policy、requestPayment、uploadId、uploads、versionId、versioning、versions、および website です。 リクエストでレスポンスヘッダーの値を上書きするクエリ文字列パラメータを指定する場合 (「GetObject」を参照) は、クエリ文字列パラメータとその値を追加します。これらのパラメータ値は署名時にエンコードする必要はありませんが、リクエストの実行時にはエンコードする必要があります。GET リクエストのクエリ文字列パラメータには、response-content-type 、response-content-language 、response-expires 、response-cache-control 、response-content-disposition 、response-content-encoding があります。 複数のオブジェクトの Delete リクエストに対して CanonicalizedResource を作成する際には、delete クエリ文字列パラメータを含める必要があります。

HTTP リクエスト URI の CanonicalizedResource の要素は、URL エンコーディングメタ文字を含め、HTTP リクエストに表示されるとおりに署名する必要があります。

CanonicalizedResource は、HTTP リクエスト URI とは異なる場合があります。特に、リクエストが HTTP Host ヘッダーを使用してバケットを指定する場合、そのバケットは HTTP リクエスト URI に表示されませんが、CanonicalizedResource にもバケットが含まれます。クエリ文字

列パラメータは、リクエスト URI に表示される可能性がありますが、CanonicalizedResource には含まれません。詳細については、「バケットの仮想ホスティング」を参照してください。

CanonicalizedAmzHeaders 要素の作成

StringToSign の CanonicalizedAmzHeaders 部分を作成するには、次の手順に従って、「x-amz-」で始まるすべての HTTP リクエストヘッダーを選択します (大文字と小文字は区別されません)。

CanonicalizedAmzHeaders プロセス

1	各 HTTP ヘッダー名を小文字に変換します。たとえば、「X-Amz-Date 」は「x-amz-date 」に変換します。
2	ヘッダーのコレクションを辞書と同じ順序でヘッダー名ごとに並べ替えます。
3	RFC 2616 のセクション 4.2 の説明に従って、同じ名前のヘッダーフィールドを 1 つの「ヘッダー名:カンマ区切りの値のリスト」にまとめます。値の間にスペースは入れません。たとえば、2 つのメタデータヘッダー「x-amz-meta-username: fred 」および「x-amz-meta-username: barney 」が 1 つのヘッダーに結合されると、「x-amz-meta-username: fred,barney 」になります。
4	(RFC 2616 のセクション 4.2 で許可されている) 複数行にわたる長いヘッダーの折り返しのスペース (改行を含む) を 1 つのスペースに置き換えて、折り返しをなくします。
5	ヘッダーのコロンの前後のスペースをすべて削除します。たとえば、ヘッダー「x-amz-meta-username: fred,barney 」は「x-amz-meta-username:fred,barney 」になります。
6	最後に、改行文字 (U+000A) を、結果の一覧の各標準化ヘッダーに追加します。この一覧のすべてのヘッダーを 1 つの文字列に連結することで、CanonicalizedResource 要素を作成します。

位置ヘッダーおよび指定 HTTP ヘッダーの StringToSign 要素

StringToSign の最初のヘッダー要素 (Content-Type、Date、および Content-MD5) は位置を示します。StringToSign には、これらのヘッダーの名前は含まれません。含まれるのはリクエストの値のみです。一方、「x-amz-」要素には名前が付いています。ヘッダーの名前と値は両方とも StringToSign に含まれます。

StringToSign の定義で呼び出される位置ヘッダーがリクエストにない場合は (たとえば、Content-Type または Content-MD5 は、PUT リクエストのオプションであり、GET リクエストには無意味です)、空の文字列 ("") をその位置に代入します。

タイムスタンプの要件

認証済みリクエストには有効なタイムスタンプ (HTTP Date ヘッダーまたは x-amz-date 代替) が必ず必要です。また、認証済みリクエストに含まれているクライアントのタイムスタンプは、リクエストの受信時の Amazon S3 システムの時刻から 15 分以内である必要があります。そうでない場合、リクエストは失敗し、RequestTimeTooSkewed エラーコードが返されます。このように制限することで、攻撃者によって傍受されたリクエストが繰り返される可能性を限定します。傍受に対する保護をさらに強化するには、認証済みリクエストに対して HTTPS 転送を使用します。

> **Note**
>
> リクエスト日の検証の制約は、クエリ文字列認証を使用しない認証済みリクエストにのみ適用されます。詳細については、「クエリ文字列による代替リクエスト認証」を参照してください。

HTTP クライアントライブラリによっては、リクエストの Date ヘッダーを設定する機能が公開されていないことがあります。標準化ヘッダーの「Date」ヘッダーの値を含めることができない場合は、代わりに「x-amz-date」ヘッダーを使用してリクエストのタイムスタンプを設定します。x-amz-date ヘッダーの値は、RFC 2616 (http://www.ietf.org/rfc/rfc2616.txt) のいずれかの形式である必要があります。x-amz-date ヘッダーがリクエストに存在する場合は、すべての Date ヘッダーがリクエスト署名の計算時に無視されます。したがって、x-amz-date ヘッダーを含める場合、Date を作成するときは、空の文字列を StringToSign に対して使用します。例については、次のセクションを参照ください。

認証の例

このセクションの例では、次の表の証明書 (実際は機能していません) を使用しています。

パラメータ	値
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecret AccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

StringToSign の例では、書式設定はそれほど複雑ではありません。\n は Unicode コードポイント U+000A を意味します。これは一般的には改行と呼ばれています。また、例では「+0000」を使用してタイムゾーンを指定しています。代わりに「GMT」を使用してタイムゾーンを指定することもできますが、署名はこの例に示したものとは異なることになります。

オブジェクト GET

この例では、awsexamplebucket1 バケットからオブジェクトを取得します。

リクエスト	StringToSign
`GET /photos/puppy.jpg HTTP/1.1` `Host: awsexamplebucket1.us-west-1.s3.amazonaws.com` `Date: Tue, 27 Mar 2007 19:36:42 +0000` `Authorization: AWS AKIAIOSFODNN7EXAMPLE:qgk2+6Sv9/oM7G3qLEjTH1a1l1g=`	`GET\n` `\n` `\n` `Tue, 27 Mar 2007 19:36:42 +0000\n` `/awsexamplebucket1/photos/puppy.jpg`

バケット名は CanonicalizedResource には含まれますが、HTTP リクエスト URI には含まれないことに注意してください。(これはホストヘッダーによって指定されます。)

> **Note**
>
> 以下の Python スクリプトは、渡されたパラメータを使用して、先程の署名を計算します。このスクリプトを使用し、必要に応じてキーおよび StringToSign を置き換えて、独自の署名を作成できます。

```
import base64
import hmac
from hashlib import sha1

access_key = 'AKIAIOSFODNN7EXAMPLE'.encode("UTF-8")
secret_key = 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'.encode("UTF-8")

string_to_sign = 'GET\n\n\nTue, 27 Mar 2007 19:36:42 +0000\n/awsexamplebucket1/
photos/puppy.jpg'.encode("UTF-8")
```

```
signature = base64.b64encode(
                                hmac.new(
                                        secret_key, string_to_sign, sha1
                                        ).digest()
                                ).strip()


print(f"AWS {access_key.decode()}:{signature.decode()}")
```

Object PUT

この例では、オブジェクトを awsexamplebucket1 バケットに配置します。

リクエスト	StringToSign
`PUT /photos/puppy.jpg HTTP/1.1` `Content-Type: image/jpeg` `Content-Length: 94328` `Host: awsexamplebucket1.s3.us-wes` `t-1.amazonaws.com` `Date: Tue, 27 Mar 2007 21:15:45 +0000` *`Authorization: AWS AKIAIOSFODNN7EXAMP`* *`LE:`* *`iqRzw+ileNPu1fhspnRs8nOjjIA=`*	`PUT\n` `\n` `image/jpeg\n` `Tue, 27 Mar 2007 21:15:45 +0000\n` `/awsexamplebucket1/photos/puppy.jpg`

リクエスト内および StringToSign 内の Content-Type ヘッダーに注意してください。また、Content-MD5 はリクエストにないので、StringToSign では空白のままであることに注意してください。

リスト

この例では、awsexamplebucket1 バケットのコンテンツのリストを取得します。

リクエスト	StringToSign
`GET /?prefix=photos&max-keys=50&marker=puppy` `HTTP/1.1`	`GET\n` `\n`

リクエスト	StringToSign
`User-Agent: Mozilla/5.0` `Host: awsexamplebucket1.s3.us-west-1.amazo` `naws.com` `Date: Tue, 27 Mar 2007 19:42:41 +0000` *`Authorization: AWS AKIAIOSFODNN7EXAMPLE:`* *`m0WP8eCtspQl5Ahe6L1SozdX9YA=`*	`\n` `Tue, 27 Mar 2007 19:42:41` `+0000\n` `/awsexamplebucket1/`

CanonicalizedResource の後ろにスラッシュがあること、そしてクエリ文字列パラメータがないことに注意してください。

Fetch

この例では、「awsexamplebucket1」バケットのアクセスコントロールポリシーのサブリソースを取得します。

リクエスト	StringToSign
`GET /?acl HTTP/1.1` `Host: awsexamplebucket1.s3.us-west-1.amazo` `naws.com` `Date: Tue, 27 Mar 2007 19:44:46 +0000` *`Authorization: AWS AKIAIOSFODNN7EXAMPLE:`* *`82ZHiFIjc+WbcwFKGUVEQspPn+0=`*	`GET\n` `\n` `\n` `Tue, 27 Mar 2007 19:44:46` `+0000\n` `/awsexamplebucket1/?acl`

サブリソースクエリ文字列パラメータがどのように CanonicalizedResource に挿入されているかに注意してください。

削除

この例では、パス形式および Date 代替を使用して、「awsexamplebucket1」バケットからオブジェクトを削除します。

リクエスト	StringToSign
`DELETE /awsexamplebucket1/photos/puppy.jpg HTTP/1.1` `User-Agent: dotnet` `Host: s3.us-west-1.amazonaws.com` `Date: Tue, 27 Mar 2007 21:20:27 +0000` `x-amz-date: Tue, 27 Mar 2007 21:20:26 +0000` *`Authorization: AWS AKIAIOSFODNN7EXAMPLE:XbyTlbQdu9Xw5o8P4iMwPktxQd8=`*	`DELETE\n` `\n` `\n` `Tue, 27 Mar 2007 21:20:26 +0000\n` `/awsexamplebucket1/photos/puppy.jpg`

代替の「x-amz-date」方式を使用して日付を指定する方法に注意してください (クライアントライブラリにより、日付を設定できない場合など)。このような場合、x-amz-date が Date ヘッダーより優先されます。このため、署名内の日付エントリに x-amz-date ヘッダーの値を含める必要があります。

アップロード

この例では、オブジェクトを、メタデータが含まれる CNAME スタイルの仮想ホストバケットにアップロードします。

リクエスト	StringToSign
`PUT /db-backup.dat.gz HTTP/1.1` `User-Agent: curl/7.15.5` `Host: static.example.com:8080` `Date: Tue, 27 Mar 2007 21:06:08 +0000` `x-amz-acl: public-read` `content-type: application/x-download` `Content-MD5: 4gJE4saaMU4BqNR0kLY+lw==` `X-Amz-Meta-ReviewedBy: joe@example.com` `X-Amz-Meta-ReviewedBy: jane@example.com` `X-Amz-Meta-FileChecksum: 0x02661779` `X-Amz-Meta-ChecksumAlgorithm: crc32`	`PUT\n` `4gJE4saaMU4BqNR0kLY+lw==\n` `application/x-download\n` `Tue, 27 Mar 2007 21:06:08 +0000\n` `x-amz-acl:public-read\n` `x-amz-meta-checksumalgorithm:crc32\n` `x-amz-meta-filechecksum:0x02661779\n` `x-amz-meta-reviewedby:joe@example.com,jane@example.com\n`

リクエスト	StringToSign
`Content-Disposition: attachment; filename=database.dat` `Content-Encoding: gzip` `Content-Length: 5913339` *`Authorization: AWS AKIAIOSFODNN7EXAMPLE:`* *`jtBQa0Aq+DkULFI8qrpwIjGEx0E=`*	`/static.example.com/db-backup.dat.gz`

「x-amz-」ヘッダーが並べ替えられ、スペースが削除されて小文字に変換されていることに注意してください。また、同じ名前を持つ複数のヘッダーが値の区切りとしてカンマを使用して結合されている様子も確認してください。

Content-Type および Content-MD5 HTTP エンティティヘッダーのみが StringToSign に表示されていることに注意してください。もう一方の Content-* エンティティヘッダーは表示されていません。

また、バケット名は CanonicalizedResource には含まれますが、HTTP リクエスト URI には含まれないことに注意してください。(これはホストヘッダーによって指定されます。)

自分のすべてのバケットのリストを取得する

リクエスト	StringToSign
`GET / HTTP/1.1` `Host: s3.us-west-1.amazonaws.com` `Date: Wed, 28 Mar 2007 01:29:59 +0000` *`Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdERIC03wnaRNKh6OqZehG9s=`*	`GET\n` `\n` `\n` `Wed, 28 Mar 2007 01:29:59 +0000\n` `/`

Unicode キー

リクエスト	StringToSign
`GET /dictionary/fran%C3%A7ais/pr` `%c3%a9f%c3%a8re HTTP/1.1` `Host: s3.us-west-1.amazonaws.com` `Date: Wed, 28 Mar 2007 01:49:49 +0000` *`Authorization: AWS AKIAIOSFODNN7EXAMP`* *`LE:DNEZGsoieTZ92F3bUfSPQcbGmlM=`*	`GET\n` `\n` `\n` `Wed, 28 Mar 2007 01:49:49 +0000\n` `/dictionary/fran%C3%A7ais/pr` `%c3%a9f%c3%a8re`

> **Note**
>
> リクエスト URI から派生した StringToSign の要素は、URL エンコーディングおよび大文字/小文字の設定を含め、文字通りに取得されます。

REST リクエストの署名に関する問題

REST リクエスト認証が失敗すると、システムは、そのリクエストに対して XML エラードキュメントで応答します。このエラードキュメントに含まれる情報は、開発者が問題を診断するのに役立ちます。特に、StringToSign エラードキュメントの SignatureDoesNotMatch 要素を確認すると、システムで使用されているリクエスト標準化が正確にわかります。

ツールキットの中には、知らないヘッダーを通知なしで事前に挿入するものがあります。たとえば、PUT の実行中にヘッダー Content-Type が追加されることがあります。この場合、挿入されたヘッダーの値は一定であることがほとんどで、Ethereal、tcpmon などのツールを使用すると、不足しているヘッダーを検出できます。

クエリ文字列による代替リクエスト認証

Authorization HTTP ヘッダーを使用する代わりに、必要な情報をクエリ文字列パラメータとして渡すことで、特定の種類のリクエストを認証できます。これは、サードパーティーのブラウザで、リクエストのプロキシを行わずにプライベートの Amazon S3 データに直接アクセスさせる場合に便利です。これを行うには、「署名付き」のリクエストを作成し、エンドユーザーのブラウザが取得できる URL としてエンコードします。さらに、署名付きのリクエストは、有効期限を指定することで制限できます。

クエリパラメータを使用するリクエストの認証の詳細については、Amazon Simple Storage Service API リファレンスの「リクエストの認証: クエリパラメータの使用 (AWS 署名バージョン 4)」を参照してください。AWS SDK を使用して署名付き URL を生成する例については、署名付き URL を使用したオブジェクトの共有 を参照してください。

署名の作成

クエリ文字列で認証済みの Amazon S3 の REST リクエストの例を次に示します。

```
GET /photos/puppy.jpg
?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Expires=1141889120&Signature=vjbyPxybdZaNmGa
%2ByT272YEAiv4%3D HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
Date: Mon, 26 Mar 2007 19:37:58 +0000
```

クエリ文字列リクエスト認証方法には、特別な HTTP ヘッダーは必要ありません。代わりに、必要な認証要素は、クエリ文字列パラメータとして指定します。

クエリ文字列パラメータ名	値の例	説明
`AWSAccessKeyId`	`AKIAIOSFODNN7EXAMPLE`	AWS アクセスキー ID。リクエストへの署名に使用する AWS シークレットアクセスキーを指定します。また、リクエストを行うデベロッパーのアイデンティティを間接的に指定します。
`Expires`	`1141889120`	署名の有効期限。エポック (1970 年 1 月 1 日の 00:00:00 UTC) からの時間が秒単位で指定されます。この時刻 (サーバーの時刻) より後に受信したリクエストは拒否されます。
`Signature`	`vjbyPxybdZaNmGa%2ByT272YEAiv4%3D`	StringToSign の HMAC-SHA1 の Base64 エンコーディングの URL エンコーディング。

クエリ文字列リクエスト認証方法は、通常の方法とは若干異なりますが、Signature リクエストパラメータおよび StringToSign 要素の形式だけが異なります。クエリ文字列リクエスト認証方法を示す疑似文法を次に示します。

```
Signature = URL-Encode( Base64( HMAC-SHA1( YourSecretAccessKey, UTF-8-Encoding-
Of( StringToSign ) ) ) );

StringToSign = HTTP-VERB + "\n" +
    Content-MD5 + "\n" +
    Content-Type + "\n" +
    Expires + "\n" +
    CanonicalizedAmzHeaders +
    CanonicalizedResource;
```

YourSecretAccessKey は、アマゾン ウェブ サービスのデベロッパーとしてサインアップするときに、Amazon によって割り当てられる AWS シークレットアクセスキー ID です。Signature が、クエリ文字列内に適切に配置されるために URL エンコードされている点に注目してください。また、HTTP では StringToSign であった位置要素が、Date では Expires に置き換えられていることに注意してください。CanonicalizedAmzHeaders および CanonicalizedResource は同じです。

> **Note**
>
> クエリ文字列認証メソッドでは、署名する文字列を計算する際に、Date または x-amz-date request ヘッダーを使用しません。

クエリ文字列リクエスト認証

リクエスト	StringToSign
`GET /photos/puppy.jpg?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&` `    Signature=NpgCjnDzrM%2BWFzoENXmpNDUsSn8%3D&` `    Expires=1175139620 HTTP/1.1`	`GET\n` `\n` `\n` `1175139620\n` `\n` `/awsexamplebucket1/photos/puppy.jpg`

リクエスト	StringToSign
`Host: awsexamplebucket1.s3.us-west-1.amazonaws.com`	

ブラウザが GET リクエストを行うとき、Content-MD5 または Content-Type ヘッダーを提供しないこと、また、x-amz- ヘッダーを設定しないことが前提となっています。したがって、`StringToSign` のこの部分は空白のままです。

Base64 エンコーディングの使用

HMAC リクエスト署名は、Base64 エンコードされている必要があります。Base64 エンコーディングでは、署名を、リクエストにアタッチできるシンプルな ASCII 文字列に変換します。プラス (+)、スラッシュ (/)、等号 (=) などの署名文字列に含めることができる文字は、URI で使用する場合には、エンコードされている必要があります。たとえば、認証コードにプラス記号 (+) が含まれる場合は、リクエストで %2B としてエンコードします。スラッシュは %2F、等号は %3D でエンコードします。

Base64 エンコーディングの例については、Amazon S3 の「認証の例」を参照してください。

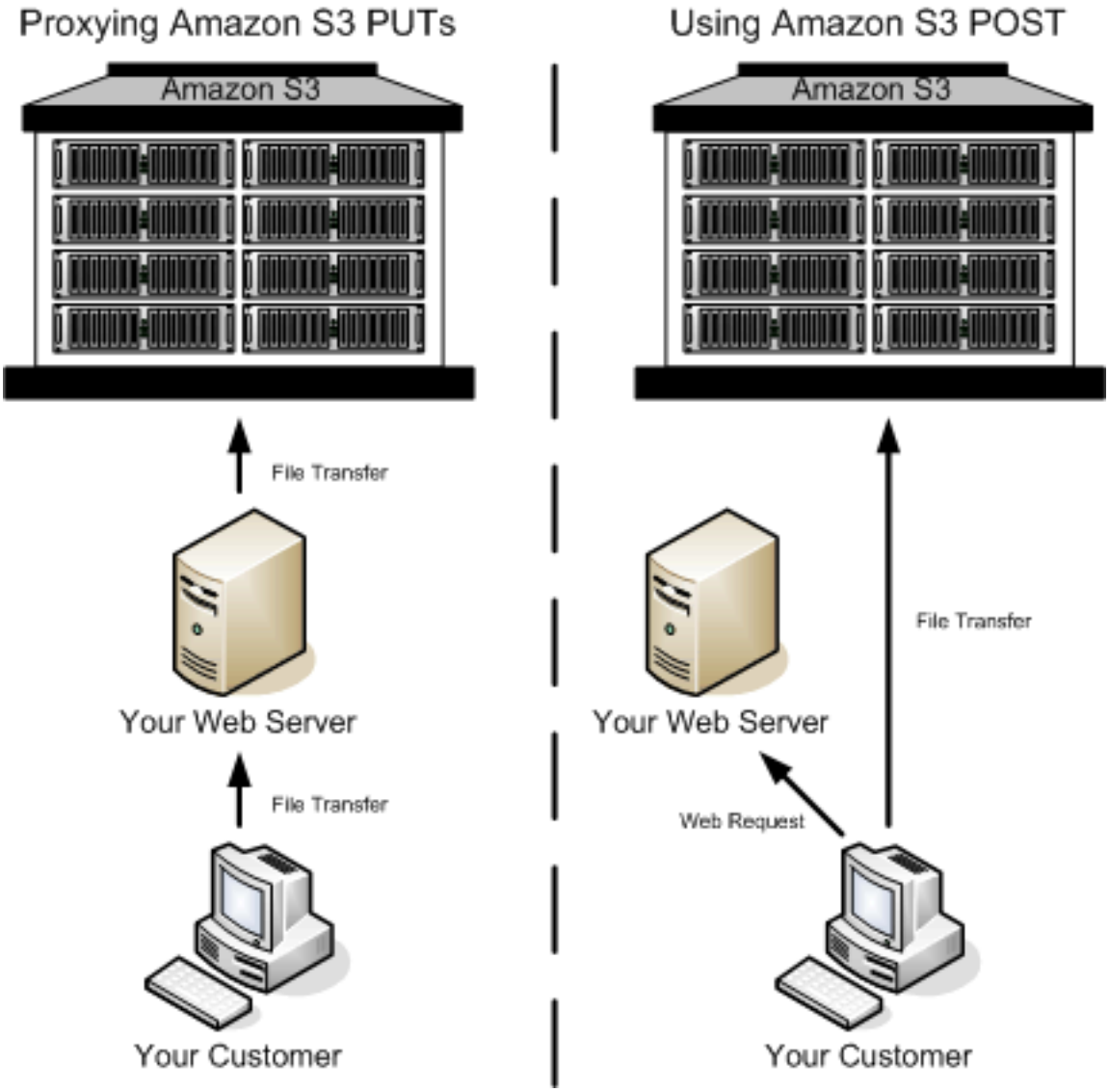
POST (AWS 署名バージョン 2) を使用したブラウザベースのアップロード

Amazon S3 では、POST がサポートされています。POST を使用すると、ユーザーがコンテンツを Amazon S3 に直接アップロードできます。POST は、アップロードを簡素化し、アップロードの待ち時間を短縮するように設計されています。また、ユーザーが Amazon S3 にデータをアップロードして保存するためのアプリケーションの費用を節約することもできます。

> **Note**
>
> このセクションで説明するリクエスト認証は、AWS 署名バージョン 2 に基づいています。これは、AWS のサービスへのインバウンドの API リクエストを認証するためのプロトコルです。
> Amazon S3 では、現在、署名バージョン 4 がサポートされています。署名バージョン 4 は、すべての AWS リージョン で AWS のサービスに対するインバウンドの API リクエストを認証するためのプロトコルです。現時点では、2014 年 1 月 30 日以前に作成された AWS リージョン は、以前のプロトコルである署名バージョン 2 のサポートを継続します。2014 年 1 月 30 日以降の新しいリージョンでは、署名バージョン 4 のみがサポートされます。そのため、これらのリージョンに対するすべてのリクエストは署名バージョン 4 で実行される必要があります。詳細については、Amazon Simple Storage Service API リファレンスの

「POST を使用したブラウザベースのアップロードでのリクエストの認証 (AWS 署名バージョン 4)」を参照してください。

次の図は、Amazon S3 の POST を使用したアップロードを示しています。



POST を使用したアップロード

1	ユーザーは、ウェブブラウザを開いて、開発者のウェブページにアクセスします。
2	ウェブページには、ユーザーがコンテンツを Amazon S3 にアップロードするために必要なすべての情報が含まれている HTTP フォームがあります。
3	ユーザーは、コンテンツを Amazon S3 に直接アップロードします。

> **Note**
>
> POST では、クエリ文字列認証がサポートされていません。

HTML フォーム (AWS 署名バージョン 2)

トピック

- HTML フォームのエンコーディング
- HTML フォーム宣言
- HTML フォームフィールド
- ポリシーの作成
- 署名の作成
- リダイレクト

Amazon S3 と通信する場合、通常は REST API または SOAP API を使用して、put、get、delete などのオペレーションを実行します。POST では、ユーザーはデータをブラウザから Amazon S3 に直接アップロードします。SOAP API の処理や REST の PUT リクエストの作成は行えません。

> **Note**
>
> SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

ユーザーがブラウザを使用してコンテンツを Amazon S3 にアップロードできるようにするには、HTML フォームを使用します。HTML フォームは、フォーム宣言とフォームフィールドで構成されます。フォーム宣言には、リクエストの概要情報が含まれています。フォームフィールドには、リクエストの詳細情報と、リクエストを認証し、指定した条件をそのリクエストが確実に満たすようにするためのポリシーが含まれています。

> **Note**
>
> フォームデータと境界 (ファイルのコンテンツは除く) は 20 KB を超えることはできません。

このセクションでは、HTML フォームを使用する方法について説明します。

HTML フォームのエンコーディング

フォームとポリシーは UTF-8 エンコーディングされている必要があります。UTF-8 エンコードをフォームに適用するには、そのエンコードを HTML 見出しで指定するかリクエストヘッダーとして指定します。

> **Note**
>
> HTML フォーム宣言は、クエリ文字列認証パラメータを受け入れません。

HTML 見出しの UTF-8 エンコードの例を次に示します。

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

リクエストヘッダーの UTF-8 エンコードの例を次に示します。

```
Content-Type: text/html; charset=UTF-8
```

HTML フォーム宣言

フォーム宣言には、アクション、メソッド、エンクロージャタイプの 3 つのコンポーネントが含まれます。これらの値が適切に設定されていないと、リクエストは失敗します。

アクションは、リクエストを処理する URL を指定します。これはバケットの URL に設定されていなければなりません。例えば、バケットの名前が `awsexamplebucket1` で、リージョンが米国西部 (北カリフォルニア) の場合、URL は `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/` になります。

> **Note**
>
> キー名はフォームフィールドで指定されます。

メソッドは POST である必要があります。

エンクロージャタイプ (enctype) を指定する必要があり、ファイルアップロードとテキストエリアアップロードの両方に対して、multipart/form-data が設定されていなければなりません。詳細については、RFC 1867 を参照してください。

Example

以下の例は、「awsexamplebucket1」バケットのフォーム宣言を示しています。

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"

enctype="multipart/form-data">
```

HTML フォームフィールド

以下の表に、HTML フォーム内で使用できるフィールドを示します。

> **Note**
>
> 変数 `${filename}` は、ユーザーによって指定されたファイル名に自動的に置き換えられ、すべてのフォームフィールドで認識されます。ブラウザまたはクライアントがファイルへの完全または部分的なパスを提供した場合、最後のスラッシュ (/) またはバックスラッシュ (\) 以降のテキストのみが使用されます。たとえば、「C:\Program Files\directory1\file.txt」は「file.txt」と解釈されます。ファイルまたはファイル名が指定されていない場合、変数は空の文字列に置き換えられます。

フィールド名	説明	必須
AWSAccessKeyId	ポリシー内の一連の制約を満たすリクエストに対して匿名ユーザーアクセスを付与するバケット所有者の AWS アクセスキー ID。リク	条件付き

フィールド名	説明	必須
	エストにポリシードキュメントが含まれる場合、このフィールドは必須です。	
acl	Amazon S3 のアクセスコントロールリスト (ACL)。無効なアクセスコントロールリストが指定されると、エラーが生成されます。ACL の詳細については、「アクセスコントロールリスト (ACL)」を参照してください。 型: 文字列 デフォルト: プライベート 有効な値: `private | public-read | public-read-write | aws-exec-read | authenticated-read | bucket-owner-read | bucket-owner-full-control`	いいえ
`Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires`	REST 固有ヘッダー。詳細については、「Put Object」を参照してください。	いいえ
key	アップロードされたキーの名前。 ユーザーによって指定されたファイル名を使用するには、${filename} 変数を使用します。たとえば、Betty がファイル lolcatz.jpg をアップロードし、開発者が /user/betty/${filename} を指定すると、そのファイルは /user/betty/lolcatz.jpg として保存されます。 詳細については、「オブジェクトメタデータの使用」を参照してください。	はい

フィールド名	説明	必須
policy	リクエストで許可されているものについて説明するセキュリティポリシー。セキュリティポリシーがないリクエストは匿名と見なされ、誰でも書き込むことができるバケットでのみ動作します。	いいえ
success_action_redirect, redirect	アップロードが成功したときにクライアントがリダイレクトされる URL。Amazon S3 は、バケット、キー、ETag の値をクエリ文字列パラメータとして URL に追加します。 success_action_redirect が指定されていない場合、Amazon S3 は、success_action_status フィールドで指定されている空のドキュメントタイプを返します。 Amazon S3 が URL を解釈できない場合、フィールドは無視されます。 アップロードが失敗した場合、Amazon S3 はエラーを表示し、ユーザーを URL にリダイレクトしません。 詳細については、「リダイレクト」を参照してください。 **Note** リダイレクトフィールド名は廃止されています。このフィールド名のサポートは削除される予定です。	いいえ

フィールド名	説明	必須
success_action_status	アップロードが正常に行われたとき、success_action_redirect が指定されていない場合にクライアントに返されるステータスコード。 有効な値は、200、201、または 204 (デフォルト) です。 値が 200 または 204 に設定されている場合、Amazon S3 は空のドキュメントとステータスコード 200 または 204 を返します。 値が 201 に設定されている場合、Amazon S3 は XML ドキュメントとステータスコード 201 を返します。XML ドキュメントの内容については、「POST Object」を参照してください。 値が設定されていないか無効な値が設定されている場合は、Amazon S3 は空のドキュメントとステータスコード 204 を返します。 **Note** Adobe Flash Player のバージョンによっては、本文が空の HTTP レスポンスが適切に処理されないことがあります。Adobe Flash でアップロードをサポートするには、success_action_status を 201 に設定することをお勧めします。	いいえ

フィールド名	説明	必須
signature	HMAC 署名は、提供された AWSAccessKeyId に対応するシークレットアクセスキーを使用して構築されます。このフィールドは、ポリシードキュメントがリクエストに含まれる場合に必要です。 詳細については、「Amazon S3 用 Identity and Access Management」を参照してください。	条件付き
x-amz-security-token	セッション認証情報で使用されるセキュリティトークン リクエストで Amazon DevPay を使用する場合は、製品トークン用とユーザートークン用の 2 つの x-amz-security-token フォームフィールドが必要です。 リクエストでセッション認証情報が使用される場合は、単一の x-amz-security-token フォームが必要です。詳細については、IAM ユーザーガイドの「IAM の一時的なセキュリティ認証情報」を参照してください。	いいえ
プレフィックスが x-amz-meta- のその他のフィールド名	ユーザーが指定したメタデータ。 Amazon S3 では、このデータは検証または使用されません。 詳細については、「PutObject」を参照してください。	いいえ

フィールド名	説明	必須
file	ファイルまたはテキストコンテンツ。 ファイルまたはコンテンツはフォームの最後のフィールドでなければなりません。その下のフィールドはすべて無視されます。 複数のファイルを一度にアップロードすることはできません。	はい

ポリシーの作成

トピック

- 有効期限
- 条件
- 条件一致
- 文字のエスケープ

ポリシーは、UTF-8 および Base64 エンコードされた JSON ドキュメントで、リクエストが満たさなければならない条件を指定します。また、コンテンツを認証するときにも使用されます。ポリシードキュメントは、どのように設計するかに応じて、アップロードごと、ユーザーごと、すべてのアップロードに対して、またはニーズに応じたその他の設計にしたがって使用できます。

> **Note**
>
> ポリシードキュメントはオプションですが、バケットを誰でも書き込むことができるようにするよりも、ポリシードキュメントを使用することを強くお勧めします。

ポリシードキュメントの例を次に示します。

```
{ "expiration": "2007-12-01T12:00:00.000Z",

  "conditions": [
```

```
    {"acl": "public-read" },

    {"bucket": "awsexamplebucket1" },

    ["starts-with", "$key", "user/eric/"],

  ]

}
```

ポリシードキュメントには、有効期限と条件が含まれます。

有効期限

有効期限要素は、ポリシーの有効期限日を ISO 8601 UTC 日付形式で指定します。たとえば「2007-12-01T12:00:00.000Z」は、2007 年 12 月 1 日の 12:00 UTC にポリシーが無効になることを指定します。ポリシーには有効期限が必ず必要です。

条件

ポリシードキュメントの条件は、アップロードされたオブジェクトのコンテンツを検証します。条件の一覧には、フォームで指定した各フォームフィールドが含まれていなければなりません (AWSAccessKeyId、署名、ポリシー、および x-ignore- プレフィックスが付いているフィールド名を除く)。

> **Note**
>
> 同じ名前のフィールドが複数ある場合、そのフィールドの値はコンマで区切る必要があります。たとえば、「x-amz-meta-tag」という名前のフィールドが 2 つあり、最初のフィールドの値が「Ninja」、2 つ目のフィールドの値が「Stallman」の場合、ポリシードキュメントは `Ninja,Stallman` に設定します。
> フォーム内のすべての変数は、ポリシーの検証前に展開されます。したがって、条件との照合は必ず、展開されたフィールドに対して行われます。たとえば、キーフィールドを `user/betty/${filename}` に設定した場合、ポリシーは `[ "starts-with", "$key", "user/betty/" ]` になる可能性があります。`[ "starts-with", "$key", "user/betty/${filename}" ]` は入力しないでください。詳細については、「条件一致」を参照してください。

以下の表にポリシードキュメント条件を示します。

要素名	説明
acl	ACL が満たす必要がある条件を指定します。 完全一致および `starts-with` をサポートします。
content-length-range	アップロードされたコンテンツの最小サイズと最大サイズを指定します。 範囲一致をサポートします。
Cache-Control、Content-Type、Content-Disposition、Content-Encoding、Expires	REST 固有ヘッダー。 完全一致および `starts-with` をサポートします。
key	アップロードされたキーの名前。 完全一致および `starts-with` をサポートします。
success_action_redirect、redirect	アップロードが成功したときにクライアントがリダイレクトされる URL。 完全一致および `starts-with` をサポートします。
success_action_status	アップロードが正常に行われたとき、success_action_redirect が指定されていない場合にクライアントに返されるステータスコード。 完全一致をサポートします。
x-amz-security-token	Amazon DevPay のセキュリティトークン。 Amazon DevPay を使用する各リクエストでは、製品トークン用とユーザートークン用の 2 つの `x-amz-security-token` フォームフィールドが必要です。結果的に、値はコンマで区切られていなければなりません。たとえば、

要素名	説明
	ユーザートークンが eW91dHViZQ== で、製品トークンが b0hnNVNKWVJIQTA= の場合、ポリシーエントリは { "x-amz-security-token": "eW91dHViZQ==,b0hnNVNKWVJIQTA=" } に設定します。
プレフィックスが x-amz-meta- のその他のフィールド名	ユーザーが指定したメタデータ。 完全一致および starts-with をサポートします。

> **Note**
>
> ツールキットにより追加のフィールドが追加されている場合は (Flash では filename が追加される)、そのフィールドをポリシードキュメントに追加する必要があります。この機能を管理できる場合は、そのフィールドの機能が Amazon S3 に無視されるようにプレフィックスの x-ignore- をフィールドに追加します。これは、そのフィールドの機能の将来のバージョンには影響しません。

条件一致

以下の表に、条件一致タイプを示します。フォーム内で指定したフォームフィールドごとに 1 つの条件を指定する必要がありますが、1 つのフォームフィールドに複数の条件を指定し、より複雑な一致条件を作成することもできます。

条件	説明
完全一致	完全一致では、フィールドが特定の値と一致するかどうかを確認します。この例は、ACL が public-read に設定される必要があることを示しています: `{"acl": "public-read" }` この例は、ACL が public-read に設定される必要があることを別の方法で示しています:

条件	説明
	`[ "eq", "$acl", "public-read" ]`
先頭が一致	先頭の値を指定する場合は、starts-with を使用します。次の例は、キーが user/betty で開始する必要があることを示しています。 `["starts-with", "$key", "user/betty/"]`
任意のコンテンツに一致	フィールド内の任意のコンテンツを許可するようにポリシーを設定するには、starts-with に空の値を使用します。次の例は、任意の success_action_redirect を許可します。 `["starts-with", "$success_action_redirect", ""]`
範囲を指定	範囲指定できるフィールドについては、範囲の上限値と下限値をコンマで区切ります。次の例は、1 ~ 10 メガバイトのサイズのファイルを許可します。 `["content-length-range", 1048579, 10485760]`

文字のエスケープ

以下の表に、ポリシードキュメント内でエスケープする必要のある文字を示します。

エスケープシーケンス	説明
\\	バックスラッシュ
\$	ドル記号

エスケープシーケンス	説明
\b	Backspace
\f	フォームフィード
\n	改行
\r	キャリッジリターン
\t	水平タブ
\v	垂直タブ
\u*xxxx*	すべての Unicode 文字

署名の作成

ステップ	説明
1	UTF-8 を使用してポリシーをエンコードします。
2	Base64 を使用して、それらの UTF-8 バイトをエンコードします。
3	HMAC SHA-1 を使用して、シークレットアクセスキーでポリシーに署名します。
4	Base64 を使用して、SHA-1 署名をエンコードします。

認証の一般的な情報については、「Amazon S3 用 Identity and Access Management」を参照してください。

リダイレクト

このセクションでは、リダイレクトを処理する方法について説明します。

一般的なリダイレクト

POST リクエストの完了時に、ユーザーは、フィールドで指定した場所にリダイレクトされます。success_action_redirectAmazon S3 が URL を解釈できない場合、success_action_redirect フィールドは無視されます。

success_action_redirect が指定されていない場合、Amazon S3 は success_action_status フィールドで指定されている空のドキュメントタイプを返します。

POST リクエストが失敗した場合、Amazon S3 はエラーを表示し、リダイレクトを行いません。

アップロード前のリダイレクト

<CreateBucketConfiguration> を使用してバケットが作成された場合、エンドユーザーをリダイレクトしなければならない可能性があります。これが発生した場合、ブラウザによっては、リダイレクトが正しく処理されないことがあります。比較的まれにですが、バケットの作成直後に発生する可能性があります。

アップロードの例 (AWS 署名バージョン 2)

トピック

- ファイルのアップロード
- テキストエリアのアップロード

> **Note**
>
> このセクションで説明するリクエスト認証は、AWS 署名バージョン 2 に基づいています。これは、AWS のサービスへのインバウンドの API リクエストを認証するためのプロトコルです。
> Amazon S3 では、現在、署名バージョン 4 がサポートされています。署名バージョン 4 は、すべての AWS リージョン で AWS のサービスに対するインバウンドの API リクエストを認証するためのプロトコルです。現時点では、2014 年 1 月 30 日以前に作成された AWS リージョン は、以前のプロトコルである署名バージョン 2 のサポートを継続します。2014 年 1 月 30 日以降の新しいリージョンでは、署名バージョン 4 のみがサポートされます。そ

のため、これらのリージョンに対するすべてのリクエストは署名バージョン 4 で実行される必要があります。詳細については、Amazon Simple Storage Service API リファレンスの「例: HTTP POST を使用したブラウザベースのアップロード (AWS 署名バージョン 4 を使用)」を参照してください。

ファイルのアップロード

この例は、ポリシーおよびフォームを作成し、添付ファイルをアップロードするプロセスを示しています。

ポリシーとフォームの作成

以下のポリシーは、Amazon S3 の awsexamplebucket1 バケットへのアップロードをサポートしています。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

このポリシーでは、以下のように規定されています。

- アップロードは、2007 年 12 月 1 日の 12:00 UTC より前に行う必要があります。
- コンテンツは awsexamplebucket1 バケットにアップロードする必要があります。
- キーの先頭は「user/eric/」でなければならなりません。
- ACL は public-read に設定されています。
- success_action_redirect が https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html に設定されています。
- オブジェクトはイメージファイルです。

- x-amz-meta-uuid タグが 14365123651274 に設定されている必要があります。
- x-amz-meta-tag には任意の値を含めることができます。

このポリシーの Base64 エンコードされたバージョンを次に示します。

```
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
```

認証情報を使用すると署名が作成されます。たとえば、0RavWzkygo6QX9caELEqKi9kDbU= は、前述のポリシードキュメントの署名です。

以下のフォームは、このポリシーを使用する DOC-EXAMPLE-BUCKET バケットへの POST リクエストをサポートします。

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
  ...
  <form action="https://DOC-EXAMPLE-BUCKET.s3.us-west-1.amazonaws.com/" method="post"
 enctype="multipart/form-data">
    Key to upload: <input type="input" name="key" value="user/eric/" /><br />
    <input type="hidden" name="acl" value="public-read" />
    <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html" />
    Content-Type: <input type="input" name="Content-Type" value="image/jpeg" /><br />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
    <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
    <input type="hidden" name="Policy" value="POLICY" />
    <input type="hidden" name="Signature" value="SIGNATURE" />
    File: <input type="file" name="file" /> <br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>
  ...
</html>
```

リクエスト例

このリクエストでは、アップロードされたイメージが 117,108 バイト (イメージデータを除く) であることを前提としています。

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
 Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: 118698

--9431149156168
Content-Disposition: form-data; name="key"

user/eric/MyPicture.jpg
--9431149156168
Content-Disposition: form-data; name="acl"

public-read
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html
--9431149156168
Content-Disposition: form-data; name="Content-Type"

image/jpeg
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

Some,Tag,For,Picture
--9431149156168
```

```
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--9431149156168
Content-Disposition: form-data; name="Policy"

eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
--9431149156168
Content-Disposition: form-data; name="Signature"

0RavWzkygo6QX9caELEqKi9kDbU=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

...file content...
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

レスポンス例

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
successful_upload.html?bucket=awsexamplebucket1&key=user/eric/
MyPicture.jpg&etag=&quot;39d459dfbc0faabbb5e179358dfb94c3&quot;
Server: AmazonS3
```

テキストエリアのアップロード

トピック

- ポリシーとフォームの作成
- リクエスト例
- レスポンス例

この例は、ポリシーおよびフォームを作成し、テキストエリアをアップロードするプロセスを示しています。テキストエリアのアップロードは、ブログ投稿のようなユーザー作成コンテンツの送信に便利です。

ポリシーとフォームの作成

以下のポリシーは、Amazon S3 の awsexamplebucket1 バケットへのテキストのアップロードをサポートしています。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
new_post.html"},
    ["eq", "$Content-Type", "text/html"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

このポリシーでは、以下のように規定されています。

- アップロードは 2007 年 12 月 1 日 12:00 GMT よりも前に行う必要があります。
- コンテンツは awsexamplebucket1 バケットにアップロードする必要があります。
- キーの先頭は「user/eric/」でなければならなりません。
- ACL は public-read に設定されています。
- success_action_redirect が https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html に設定されています。
- オブジェクトは HTML テキストです。
- x-amz-meta-uuid タグが 14365123651274 に設定されている必要があります。
- x-amz-meta-tag には任意の値を含めることができます。

このポリシーの Base64 エンコードされたバージョンを次に示します。

```
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXR
```

```
pb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obnNtaXRoIn0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCAidXNlci
LAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3QiOiAiaHR0cDovL2p
C5zMy5hbWF6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSIsICJ0ZXh0L2h0
CAgIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiR4LWFtei1t
IsICIiXQogIF0KfQo=
```

認証情報を使用すると署名が作成されます。たとえば、qA7FWXKq6VvU68lI9KdveT1cWgF= は、前述のポリシードキュメントの署名です。

以下のフォームは、このポリシーを使用する DOC-EXAMPLE-BUCKET バケットへの POST リクエストをサポートします。

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
  ...
  <form action="https://DOC-EXAMPLE-BUCKET.s3.us-west-1.amazonaws.com/" method="post"
 enctype="multipart/form-data">
    Key to upload: <input type="input" name="key" value="user/eric/" /><br />
    <input type="hidden" name="acl" value="public-read" />
    <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html" />
    <input type="hidden" name="Content-Type" value="text/html" />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
    <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
    <input type="hidden" name="Policy" value="POLICY" />
    <input type="hidden" name="Signature" value="SIGNATURE" />
    Entry: <textarea name="file" cols="60" rows="10">

Your blog post goes here.

    </textarea><br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>
  ...
</html>
```

リクエスト例

このリクエストでは、アップロードされたイメージが 117,108 バイト (イメージデータを除く) であることを前提としています。

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
 Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=178521717625888
Content-Length: 118635

-178521717625888
Content-Disposition: form-data; name="key"

ser/eric/NewEntry.html
--178521717625888
Content-Disposition: form-data; name="acl"

public-read
--178521717625888
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html
--178521717625888
Content-Disposition: form-data; name="Content-Type"

text/html
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-tag"

Interesting Post
--178521717625888
```

```
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--178521717625888
Content-Disposition: form-data; name="Policy"
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN

--178521717625888
Content-Disposition: form-data; name="Signature"

qA7FWXKq6VvU68lI9KdveT1cWgF=
--178521717625888
Content-Disposition: form-data; name="file"

...content goes here...
--178521717625888
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--178521717625888--
```

レスポンス例

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html?
bucket=awsexamplebucket1&key=user/eric/
NewEntry.html&etag=40c3271af26b7f1672e41b8a274d28d4
Server: AmazonS3
```

Adobe Flash での POST

このセクションでは、Adobe Flash で POST を使用する方法について説明します。

Adobe Flash Player のセキュリティ

デフォルトでは、Adobe Flash Player のセキュリティモデルにより、Adobe Flash Player が、SWF ファイルを提供するドメイン外のサーバーにネットワーク接続することはできません。

このデフォルトの設定を上書きするには、公開されている crossdomain.xml ファイルを、POST アップロードを受け入れるバケットにアップロードする必要があります。crossdomain.xml ファイルの例を次に示します。

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" secure="false" />
</cross-domain-policy>
```

> **Note**
>
> Adobe Flash セキュリティモデルの詳細については、Adobe のウェブサイトを参照してください。
> crossdomain.xml ファイルをバケットに追加すると、Adobe Flash Player はバケット内の crossdomain.xml に接続できるようになります。ただし、実際の Amazon S3 バケットへのアクセス許可は付与されません。

Adobe Flash に関する考慮事項

Adobe Flash の FileReference API により、`Filename` フォームフィールドが POST リクエストに追加されます。FileReference API アクションを使用して Amazon S3 にアップロードを行う Adobe Flash アプリケーションを作成する場合は、次の条件をポリシーに含めます。

```
['starts-with', '$Filename', '']
```

Adobe Flash Player のバージョンによっては、本文が空の HTTP レスポンスが適切に処理されないことがあります。本文が空でないレスポンスを返すように POST を設定するには、`success_action_status` を 201 に設定します。これにより、Amazon S3 はステータスコード 201 で XML ドキュメントを返します。XML ドキュメントの内容については、「POST Object」を参照してください。フォームフィールドの詳細については、「HTML フォームフィールド」を参照してください。

設計パターンのベストプラクティス: Amazon S3 のパフォーマンスの最適化

Amazon S3 のストレージに対してアップロードおよび取得を行う際に、アプリケーションはリクエストのパフォーマンスとして 1 秒あたり数千のトランザクションを容易に達成できます。Amazon S3 は、高いリクエストレートに自動的にスケールされます。例えば、アプリケーションは、パーティショニングされた Amazon S3 プレフィックスごとに毎秒 3,500 回以上の PUT/COPY/POST/DELETE リクエストまたは 5,500 回以上の GET/HEAD リクエストを達成できます。バケット内のプレフィックスの数に制限はありません。並列化を使用することによって、読み取りまたは書き込みのパフォーマンスを向上させることができます。例えば、Amazon S3 バケットに 10 個のプレフィックスを作成して読み取りを並列化すると、読み取りパフォーマンスを 1 秒あたり 55,000 回の読み取りリクエストにスケールできます。同様に、複数のプレフィックスに書き込むことで、書き込みオペレーションをスケールできます。読み取りオペレーションと書き込みオペレーションの両方の場合、スケーリングは瞬時にではなく段階的に行われます。Amazon S3 が新たに高くなったリクエストレートに合わせてスケーリングしている間に、503 (Slow Down) エラーが表示される場合があります。これらのエラーは、スケーリングが完了すると解消されます。プレフィックスの作成と使用の詳細については、「プレフィックスを使用してオブジェクトを整理する」を参照してください。

Amazon S3 上のデータレイクアプリケーションによっては、ペタバイトを超えるデータに対して実行されるクエリで数百万から数十億のオブジェクトをスキャンします。これらのデータレイクアプリケーションは、Amazon EC2 インスタンスのネットワークインターフェイスの使用を最大限に高めて、単一のインスタンスで最大 100 Gb/s の転送レートを実現しています。その後、これらのアプリケーションは、複数のインスタンスにわたってスループットを集約して 1 秒あたり複数テラバイトを確保します。

ソーシャルメディアメッセージングアプリケーションなどの他のアプリケーションは、レイテンシーの影響を受けやすいアプリケーションです。このようなアプリケーションでは、小さなオブジェクト (大きなオブジェクトの場合は最初のバイトを受け取るまで) のレイテンシーで約 100～200 ミリ秒の一定のレイテンシーを実現できます。

他の AWS のサービスもさまざまなアプリケーションアーキテクチャのパフォーマンスの高速化に役立ちます。例えば、HTTP 接続ごとの転送レートを高めたい場合やレイテンシーをミリ秒単位に抑えたい場合は、Amazon CloudFront または Amazon ElastiCache を Amazon S3 のキャッシュとして使用します。

また、長距離間のクライアントと S3 バケットのデータ転送を高速化する場合は、Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定 を使用します。Transfer

Acceleration は、CloudFront の世界中に点在するエッジロケーションを使用して、長距離間のデータ転送を高速化します。Amazon S3 ワークロードが AWS KMS によるサーバー側の暗号化を使用している場合、ユースケースでサポートされるリクエスト率については、『AWS Key Management Service デベロッパーガイド』の「AWS KMS 制限」を参照してください。

以下のトピックでは、Amazon S3 を使用するアプリケーションのパフォーマンスを最適化するためのベストプラクティスガイドラインと設計パターンについて説明します。Amazon S3 のパフォーマンスの最適化に関する最新情報については、「Amazon S3 のパフォーマンスのガイドライン」および「Amazon S3 のパフォーマンスの設計パターン」を参照してください。

> **Note**
>
> Amazon S3 Express One Zone ストレージクラスをディレクトリバケットで使用する方法の詳細については、「S3 Express One Zone とは」と「ディレクトリバケット」を参照してください。

トピック

- Amazon S3 のパフォーマンスのガイドライン
- Amazon S3 のパフォーマンスの設計パターン

Amazon S3 のパフォーマンスのガイドライン

Amazon S3 に対してオブジェクトのアップロードや取得を行うアプリケーションを作成する場合は、ベストプラクティスガイドラインに従ってパフォーマンスを最適化してください。また、より詳細な「パフォーマンス設計パターン」も提供しています。

アプリケーションで Amazon S3 の最適なパフォーマンスを得るために、以下のガイドラインを推奨しています。

トピック

- パフォーマンスを測定する
- ストレージ接続を水平にスケールする
- バイト範囲のフェッチを使用する

- レイテンシーの影響を受けやすいアプリケーションのリクエストを再試行する
- 同じ AWS リージョン で Amazon S3 (ストレージ) と Amazon EC2 (コンピューティング) を組み合わせる
- Amazon S3 Transfer Acceleration を使用して距離によるレイテンシーを最小限に抑える
- 最新バージョンの AWS SDK を使用する

パフォーマンスを測定する

パフォーマンスを測定する際、ネットワークスループット、CPU、および DRAM の要件を確認してください。これらのさまざまなリソースの要件の組み合わせに応じて、さまざまな Amazon EC2 インスタンスタイプを評価することをお勧めします。インスタンスタイプの詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスタイプ」を参照してください。

これは、パフォーマンスの測定時に HTTP 分析ツールを使用して DNS ルックアップ時間、レイテンシー、およびデータ転送速度を確認する際にも役立ちます。

パフォーマンス要件を理解し、アプリケーションのパフォーマンスを最適化するために、受け取った 503 エラーレスポンスをモニタリングすることもできます。特定のパフォーマンスメトリクスのモニタリングには、追加費用が発生する場合があります。詳細については、「Amazon S3 の料金」を参照してください。

503 (Slow Down) ステータスエラーレスポンスの数のモニタリング

503 のステータスエラーレスポンス数をモニタリングするには、次のいずれかのオプションを使用できます。

- Amazon S3 に対する Amazon CloudWatch リクエストメトリクスの使用 CloudWatch リクエストメトリクスには、5xx ステータスレスポンスのメトリクスが含まれています。CloudWatch リクエストメトリクスの詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。
- Amazon S3 ストレージレンズの [高度なメトリクス] セクションにある 503 (Service Unavailable) エラー数を使用してください。詳細については、「S3 ストレージレンズのメトリクスを使用してパフォーマンスを改善する」を参照してください。
- Amazon S3 サーバーアクセスロギングの使用 サーバーアクセスログを使用すると、503 (Internal Error) レスポンスを受け取ったすべてのリクエストをフィルタリングして確認できます。Amazon Athena を使用して、ログを解析することもできます。サーバーアクセスログ記録の詳細については、「サーバーアクセスログによるリクエストのログ記録」を参照してください。

HTTP 503 ステータスエラーコードの数をモニタリングすることで、どのプレフィックス、キー、またはバケットが最もスロットリングリクエストを受け取っているかについて、貴重な分析情報を得られることがよくあります。

ストレージ接続を水平にスケールする

多くの接続間にリクエストを分散することが、パフォーマンスを水平にスケールする一般的な設計パターンです。パフォーマンスの高いアプリケーションを作成するには、Amazon S3 を従来のストレージサーバーのような 1 つのネットワークエンドポイントではなく、非常に大きな分散システムと考えます。最適なパフォーマンスは、複数の同時リクエストを Amazon S3 に発行することで実現できます。これらのリクエストを別々の接続に分散して、Amazon S3 の利用可能な帯域幅を最大化します。Amazon S3 には、バケットへの接続数に制限はありません。

バイト範囲のフェッチを使用する

GetObject リクエストで HTTP ヘッダーの `Range` を使用すると、オブジェクトのバイト範囲をフェッチして指定した部分のみを転送できます。Amazon S3 への同時接続を使用して、同じオブジェクトのさまざまなバイト範囲をフェッチできます。これにより、単一のオブジェクト全体のリクエストに対して高い集約スループットを実現できます。大きなオブジェクトの小さい範囲をフェッチすると、リクエストの中断時にアプリケーションで再試行回数の向上が可能になります。詳細については、「オブジェクトのダウンロード」を参照してください。

バイト範囲リクエストの標準的なサイズは 8 MB または 16 MB です。マルチパートアップロードを使用してオブジェクトを PUT する場合、最適なパフォーマンスのために同じパートサイズで (少なくともパート境界に沿って整列されている) それらのオブジェクトを GET することをお勧めします。GET リクエストは、個々のパートを直接アドレス指定できます (例えば、`GET ? partNumber=N.`)。

レイテンシーの影響を受けやすいアプリケーションのリクエストを再試行する

積極的なタイムアウトと再試行は、一定のレイテンシーの促進に役立ちます。Amazon S3 を大規模に利用している場合、最初のリクエストが遅くても、再試行のリクエストでは別のパスを選択してすぐに成功することがあります。AWS SDK には、特定のアプリケーションの許容値に調整できる設定可能なタイムアウト値と再試行値があります。

同じ AWS リージョン で Amazon S3 (ストレージ) と Amazon EC2 (コンピューティング) を組み合わせる

S3 のバケット名はグローバルに一意ですが、各バケットはバケットの作成時に選択したリージョンに保存されます。パフォーマンスを最適化するには、可能であれば、同じ AWS リージョン の Amazon EC2 インスタンスからバケットにアクセスすることをお勧めします。これにより、ネットワークレイテンシーとデータ転送費用を低減できます。

データ転送費用の詳細については、「Amazon S3 の料金」を参照してください。

Amazon S3 Transfer Acceleration を使用して距離によるレイテンシーを最小限に抑える

Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定 は、クライアントと S3 のバケットの長距離間の高速、簡単、安全なファイル転送を管理します。Transfer Acceleration は、Amazon CloudFront の世界中に点在するエッジロケーションを利用します。エッジロケーションに到着したデータは、最適化されたネットワークパスで Amazon S3 にルーティングされます。Transfer Acceleration は、大陸間で定期的にギガバイトからテラバイト単位のデータを転送するのに最適です。また、中央のバケットに対して世界中のお客様からアップロードが行われるクライアントにも役立ちます。

Amazon S3 Transfer Acceleration の速度比較ツールを使用すると、高速化した場合と高速化していない場合の Amazon S3 リージョン間でのアップロード速度を比較できます。速度比較ツールでは、マルチパートアップロードを使用して、ブラウザからさまざまな Amazon S3 のリージョンへのファイル転送を行い、Amazon S3 Transfer Acceleration を使用した場合と使用していない場合の比較が行われます。

最新バージョンの AWS SDK を使用する

AWS SDK では、Amazon S3 のパフォーマンスの最適化のために推奨されている多くのガイドラインが組み込みでサポートされています。SDK では、アプリケーションから Amazon S3 を利用するためのシンプルな API が提供されており、最新のベストプラクティスに従うように定期的に更新されます。例えば、SDK には、HTTP 503 エラーでリクエストを自動的に再試行するロジックが含まれており、遅い接続に応答して適用するためにコードに投資しています。

また、SDK では Transfer Manager も提供されています。これは、接続の水平スケーリングを自動化し、必要に応じてバイト範囲のリクエストを使用して 1 秒あたりに数千ものリクエストを実現しま

す。最新バージョンの AWS SDK を使用して最新のパフォーマンス最適化機能を取得することが重要です。

また、HTTP REST API リクエストを使用している場合は、パフォーマンスを最適化することもできます。REST API を使用する際、SDK の一部である同じベストプラクティスに従う必要があります。オブジェクトデータを同時フェッチできるように、リクエストが遅い場合のタイムアウトと再試行、および複数の接続を可能にしてください。REST API の使用については、Amazon Simple Storage Service API Reference を参照してください。

Amazon S3 のパフォーマンスの設計パターン

Amazon S3 に対してオブジェクトのアップロードや取得を行うアプリケーションを設計する場合は、アプリケーションの最適なパフォーマンスを実現するためにベストプラクティスの設計パターンを使用してください。また、アプリケーションのアーキテクチャの計画時に考慮すべきパフォーマンスガイドラインも提供しています。

パフォーマンスを最適化するには、以下の設計パターンを使用します。

トピック

- 頻繁にアクセスされるコンテンツにキャッシュを使用する
- レイテンシーの影響を受けやすいアプリケーションのタイムアウトと再試行
- 高スループットのための水平スケーリングとリクエスト並列化
- Amazon S3 Transfer Acceleration を使用して長距離間のデータ転送を高速化する

頻繁にアクセスされるコンテンツにキャッシュを使用する

Amazon S3 にデータを保存するアプリケーションの多くは、ユーザーから繰り返しリクエストされる「作業セット」と言えるデータを提供します。ワークロードで一連のよく使用されるオブジェクトに対して繰り返し GET リクエストを送信する場合は、Amazon CloudFront、Amazon ElastiCache、AWS Elemental MediaStore などのキャッシュを使用してパフォーマンスを最適化することができます。キャッシュ導入が成功すると、レイテンシーが低くなり、データ転送速度が速くなります。また、アプリケーションでキャッシュを使用すると Amazon S3 にリクエストを直接送信する回数も減るため、リクエストにかかる費用を削減できます。

Amazon CloudFront は、各地に点在する一連の大規模な POP (Point Of Presence) で Amazon S3 のデータを透過的にキャッシュする高速なコンテンツ配信ネットワーク (CDN) です。複数のリージョ

ンまたはインターネットからオブジェクトにアクセスする場合に CloudFront を使用すると、オブジェクトにアクセスするユーザーの近くにデータをキャッシュできます。これにより、Amazon S3 のアクセス数の多いコンテンツの配信パフォーマンスを高めることができます。CloudFront の詳細については、Amazon CloudFront 開発者ガイドを参照してください。

Amazon ElastiCache は、マネージド型のインメモリキャッシュです。ElastiCache を使用すると、オブジェクトをメモリにキャッシュする Amazon EC2 インスタンスをプロビジョニングできます。このキャッシュにより、GET レイテンシーが数桁減少し、ダウンロードスループットが大幅に向上します。ElastiCache を使用するには、アプリケーションのロジックを変更して、アクセス数の多いオブジェクトをキャッシュに保存し、Amazon S3 にそのオブジェクトをリクエストする前にキャッシュを確認するようにします。ElastiCache を使用して Amazon S3 の GET のパフォーマンスを向上させる例については、ブログ記事の「Turbocharge Amazon S3 with Amazon ElastiCache for Redis」を参照してください。

AWS Elemental MediaStore は、Amazon S3 の動画ワークフローとメディア配信のために特別に作成されたキャッシュおよびコンテンツ配信システムです。MediaStore には、動画専用のエンドツーエンドのストレージ API が用意されており、パフォーマンスが重視される動画ワークロードに最適です。MediaStore の詳細については、「AWS Elemental MediaStore ユーザーガイド」を参照してください。

レイテンシーの影響を受けやすいアプリケーションのタイムアウトと再試行

アプリケーションが Amazon S3 から再試行が必要なことを示すレスポンスを受け取る場合があります。Amazon S3 は、バケット名とオブジェクト名を関連するオブジェクトデータにマッピングします。アプリケーションで発生するリクエスト率が高い場合 (通常、少数のオブジェクトに対して 1 秒あたり 5,000 リクエストを超える率が持続される)、アプリケーションは HTTP 503 slowdown レスポンスを受信することがあります。これらのエラーが発生した場合、各 AWS SDK はエクスポネンシャルバックオフを使用して自動再試行ロジックを実装します。AWS SDK を使用していない場合は、HTTP 503 エラーの受信時に再試行ロジックを実装する必要があります。バックオフテクニックの詳細については、「Amazon Web Services 全般のリファレンス」の「AWS でのエラー再試行とエクスポネンシャルバックオフ」を参照してください。

Amazon S3 は、処理を継続するための新しいリクエストレートに応じて自動的にスケールし、パフォーマンスを動的に最適化します。Amazon S3 が新しいリクエスト率のために内部的に最適化している間、最適化が完了するまで一時的に HTTP 503 リクエストレスポンスが送信されます。Amazon S3 が新しいリクエストレートに応じてパフォーマンスを内部的に最適化すると、リクエストはすべて再試行なしで通常どおり処理されます。

レイテンシーが重要なアプリケーションの場合、Amazon S3 では遅いオペレーションを追跡して積極的に再試行することをお勧めします。リクエストを再試行する際は、Amazon S3 に新しく接続して改めて DNS ルックアップを実行することをお勧めします。

可変サイズの大きいリクエスト (例えば、128 MB 超) を実行する際、達成されるスループットを追跡し、リクエストのうち、遅い方から 5 パーセントを再試行することをお勧めします。小さいリクエスト (例えば、512 KB 未満) を実行する際、レイテンシーの中央値が数十ミリ秒の範囲内であることが多い場合、2 秒後に GET または PUT オペレーションを再試行することをお勧めします。追加の再試行が必要な場合のベストプラクティスはバックオフすることです。例えば、2 秒後に 1 回目の再試行を発行し、さらに 4 秒後に 2 回目の再試行を発行することをお勧めします。

アプリケーションが Amazon S3 に固定サイズのリクエストを実行する場合は、それぞれのリクエストの応答時間はより一定になると考えられます。この場合、シンプルな戦略はリクエストのうち、遅い方から 1 パーセントを特定してそれらを再試行することです。1 回の再試行でもレイテンシーの低減において効果的でありことが多いです。

サーバー側の暗号化で AWS Key Management Service (AWS KMS) を使用している場合、ユースケースでサポートされるリクエスト率については、AWS Key Management Service デベロッパーガイドの「制限」を参照してください。

高スループットのための水平スケーリングとリクエスト並列化

Amazon S3 は大規模な分散システムです。その規模を活用できるように、並列リクエストを Amazon S3 のサービスエンドポイントに水平にスケールすることをお勧めします。このようなスケーリングのアプローチは、Amazon S3 でのリクエストの分散だけでなく、ネットワークで複数のパスに負荷を分散するためにも役立ちます。

転送のスループットを高めるために、Amazon S3 では複数の接続によってデータの GET や PUT を並列で行うアプリケーションを使用することをお勧めします。このような並列化は、AWS Java SDK の Amazon S3 Transfer Manager でサポートされています。また、その他のほとんどの AWS SDK でも同様の機能が提供されています。一部のアプリケーションでは、さまざまなアプリケーションスレッドで、またはさまざまなアプリケーションインスタンスで複数のリクエストを同時に起動することで、並列接続を実現できます。採用する最良のアプローチは、アプリケーション、およびアクセスするオブジェクトの構造によって異なります。

AWS SDK を使用して、AWS SDK で転送の管理を使用するのではなく GET リクエストまたは PUT リクエストを直接発行できます。このアプローチにより、ワークロードをより直接的に調整できると同時に、発生する可能性がある HTTP 503 レスポンスの再試行とその処理に引き続き SDK のサポートを活用できます。一般的なルールとして、リージョン内の大きなオブジェクトを Amazon S3 から

Amazon EC2 にダウンロードする場合は、8 ~ 16 MB の粒度でオブジェクトのバイト範囲の同時リクエストを実行することをお勧めします。同時リクエストは、必要なネットワークスループットの 85 ~ 90 MB/秒ごとに 1 つ実行します。10 Gb/s のネットワークインターフェイスカード (NIC) を使用するには、個別の接続で約 15 の同時リクエストを使用します。より多くの接続で同時リクエストをスケールアップして、25 Gb/s や 100 Gb/s の NIC などのより高速な NIC を使用できます。

パフォーマンスの測定は、同時に発行するリクエスト数を調整する場合に重要です。一度に 1 つのリクエストから始めることをお勧めします。達成されるネットワーク帯域幅とデータの処理でアプリケーションで使用されるその他のリソースの使用を測定します。その後、ボトルネックとなっているリソース (つまり、使用量が最も高いリソース) と有用である可能性が高いリクエスト数を特定できます。例えば、一度に 1 つのリクエストの処理で CPU 使用率が 25 パーセントの場合、これは最大 4 つの同時リクエストに対応できることを示しています。測定は不可欠であり、リクエスト率としてのリソース利用が向上していることを確認する価値があります。

アプリケーションで REST API を使用して Amazon S3 にリクエストを直接発行する場合は、HTTP 接続のプールを使用して、一連のリクエストで各接続を再利用することをお勧めします。リクエストごとの接続セットアップを回避すると、各リクエストで TCP スロースタートと Secure Sockets Layer (SSL) ハンドシェイクを実行する必要がなくなります。REST API の使用については、Amazon Simple Storage Service API Reference を参照してください。

最後に、DNS に注意するとともに、リクエストが Amazon S3 の広範な IP アドレスのプールに分散されていることを確認することもお勧めします。Amazon S3 の DNS の問い合わせは、多数の IP エンドポイントのリストを確認します。ただし、キャッシュリゾルバ、または 1 つの IP アドレスを再利用するアプリケーションコードでは、アドレス多様性とそれによる負荷分散のメリットが得られません。コマンドラインツールの `netstat` などのネットワークユーティリティツールを使用すると、Amazon S3 との通信に使用されている IP アドレスを確認できます。また、使用する DNS 設定のガイドラインも提供しています。このガイドラインの詳細については、「リクエストの実行」を参照してください。

Amazon S3 Transfer Acceleration を使用して長距離間のデータ転送を高速化する

Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定 は、世界中のクライアントと Amazon S3 を使用する特定のリージョンのアプリケーションの距離によるレイテンシーを最小限に抑える、またはなくすために有効です。Transfer Acceleration は、データの転送に CloudFront の世界中に点在するエッジロケーションを利用します。AWS エッジネットワークでは、50 を超えるロケーションに接続ポイントがあります。現在、このネットワークは、CloudFront

でのコンテンツの配信や Amazon Route 53 に対する DNS の問い合わせへの迅速な応答のために使用されています。

このエッジネットワークは、Amazon S3 との間のデータ転送の高速化にも役立ちます。これは、大陸全域または大陸間でデータを転送する、高速なインターネット接続がある、大きなオブジェクトを使用する、またはアップロードするコンテンツが多数あるアプリケーションに最適です。エッジロケーションに到着したデータは、最適化されたネットワークパスで Amazon S3 にルーティングされます。一般的に、Amazon S3 のリージョンから遠いほど、Transfer Acceleration による速度の向上が期待できます。

新しいバケットまたは既存のバケットで Transfer Acceleration をセットアップできます。離れた Amazon S3 Transfer Acceleration エンドポイントを使用して、AWS のエッジロケーションを使用することができます。Transfer Acceleration がクライアントのリクエストのパフォーマンスに役立つかどうかをテストする最も良い方法は、Amazon S3 Transfer Acceleration の速度比較ツールを使用することです。ネットワークの設定および条件は、随時、場所によって異なります。そのため、Amazon S3 Transfer Acceleration がアップロードのパフォーマンスを向上させることができると考えられる転送に対してのみ料金が発生します。さまざまな AWS SDK での Transfer Acceleration の使用については、「S3 Transfer Acceleration の有効化と使用」を参照してください。

Amazon S3 on Outposts とは

AWS Outposts は、同じ AWS インフラストラクチャ、AWS サービス、API、およびツールを実質的にあらゆるデータセンター、コロケーションスペース、またはオンプレミスの施設に提供するフルマネージドサービスであり、真に一貫性のあるハイブリッドエクスペリエンスを実現します。AWS Outposts は、オンプレミスシステムへの低レイテンシーアクセス、ローカルデータ処理、データレジデンシー、およびローカルシステムの相互依存性を持つアプリケーションの移行に必要なワークロードに最適です。詳細については、『AWS Outposts ユーザーガイド』の「What is AWS Outposts? (とは？)」 を参照してください。

Amazon S3 on Outposts を使用すると、S3 バケットを Outposts に作成して、オンプレミスでのオブジェクトの保存と取得を容易に行うことができます。S3 on Outposts は、OUTPOSTS という新しいストレージクラスを提供し、これは Amazon S3 API を使用し、Outposts の複数のデバイスとサーバー間でデータを永続的かつ冗長的に保存するように設計されています。仮想プライベートクラウド (VPC) を介したアクセスポイントとエンドポイント接続を使用して、Outposts バケットと通信します。

Outposts バケットでは、Amazon S3 と同じ API と機能 (アクセスポリシー、暗号化、タグ付けなど) を使用できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。

- S3 on Outposts のしくみ
- S3 on Outposts の機能
- 関連サービス
- S3 on Outposts へのアクセス
- S3 on Outposts の支払い
- 次のステップ

S3 on Outposts のしくみ

S3 on OOutposts は、オブジェクトストレージサービスであり、データを Outpost 上のバケット内にオブジェクトとして保存します。オブジェクトとは、データファイルと、そのファイルを記述する任意のメタデータのことです。バケット とは、オブジェクトのコンテナのことです。

S3 on Outposts にデータを保存するには、まずバケットを作成します。バケットを作成するときには、バケット名とバケットを保持する Outpost を指定します。S3 on Outposts バケットにアクセス

してオブジェクト操作を実行するには、次にアクセスポイントを作成して設定します。アクセスポイントにリクエストをルーティングするエンドポイントも作成する必要があります。

アクセスポイントは、S3 にデータを保存するあらゆる AWS のサービス やお客様のアプリケーションのデータアクセスを簡素化します。アクセスポイントは、バケットにアタッチされた名前付きのネットワークエンドポイントであり、`GetObject` や `PutObject` などのオブジェクト操作を実行するために使用できます。各アクセスポイントには、個別の許可とネットワーク制御があります。

AWS Management Console、AWS CLI、AWS SDK、または REST API を使用して、S3 on Outposts バケット、アクセスポイント、およびエンドポイントの作成と管理ができます。S3 on Outposts バケット内のオブジェクトをアップロードおよび管理するには、AWS CLI、AWS SDK、または REST API を使用できます。

リージョン

AWS Outposts プロビジョニング時、ユーザーまたは AWS は、バケット操作とテレメトリ用に、Outpost を選択した AWS リージョン または Outposts ホームリージョンに接続するサービスリンク接続を作成します。Outpost は、親 AWS リージョン への接続性に依存します。Outposts ラックは、切断された操作や接続がない環境向けに設計されていません。詳細については、「AWS Outposts ユーザーガイド」の「AWS リージョン への Outpost の接続性」を参照してください。

バケット

バケットとは、S3 on Outposts に保存されるオブジェクトのコンテナです。バケットにはオブジェクトをいくつでも保存でき、1 つの Outpost には 1 つのアカウントにつき最大 100 個のバケットを保存できます。

バケットを作成するときには、バケット名を入力し、バケットが存在する Outpost を選択します。バケットの作成後は、バケット名を変更したり、バケットを別の Outpost に移動したりすることはできません。バケット名は、Amazon S3 バケットの命名規則に従う必要があります。S3 on Outposts では、バケット名は Outpost と AWS アカウント に対して一意です、S3 on Outposts バケットには、バケットを識別するための `outpost-id`、`account-id`、およびバケット名が必要です。

以下の例では、S3 on Outposts バケットの Amazon リソースネーム (ARN) 形式を示しています。ARN は、Outpost が属するリージョン、Outpost アカウント、Outpost ID、およびバケット名で構成されます。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/bucket/bucket-name
```

すべてのオブジェクトはバケット内に保存されます。Outposts バケット内の任意のオブジェクトにアクセスするには、アクセスポイントを使用する必要があります。オブジェクト操作のためにバケットを指定するときには、アクセスポイント ARN またはアクセスポイントエイリアスを使用します。アクセスポイントエイリアスの詳細については、「S3 on Outposts アクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

次の例は、`outpost-id`、`account-id`、アクセスポイント名を含む S3 on Outposts のアクセスポイント ARN 形式を示しています。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

バケットの詳細については、「S3 on Outposts バケットの操作」を参照してください。

オブジェクト

オブジェクトは、S3 on Outposts に保存される基本エンティティです。オブジェクトは、オブジェクトデータとメタデータで構成されます。メタデータは、オブジェクトを表現する名前と値のペアのセットです。これには最終更新日などのデフォルトメタデータや、`Content-Type` などの標準 HTTP メタデータが含まれます。また、オブジェクトの保存時にカスタムメタデータを指定することもできます。オブジェクトは、キー (または名前) によってバケット内で一意に識別されます。

Amazon S3 on Outposts では、オブジェクトデータは常に Outpost に保存されます。AWS が Outpost ラックを設置すると、データの常駐要件を満たすために、データは Outpost にローカルに保たれます。オブジェクトが Outpost を離れたり、AWS リージョン 外に出たりすることはありません。AWS Management Consoleはリージョン内でホストされるため、コンソールを使用して Outpost にオブジェクトをアップロードしたり、管理したりすることはできません。ただし REST API、AWS Command Line Interface (AWS CLI)、および AWS SDK を使用して、アクセスポイントを介してオブジェクトのアップロードと管理を行うことができます。

キー

オブジェクトキー (または キー名) は、バケット内のオブジェクトの固有の識別子です。バケット内のすべてのオブジェクトは、厳密に 1 個のキーを持ちます。バケットとオブジェクトキーの組み合わせで、各オブジェクトを一意に識別します。

次の例は、S3 on Outposts オブジェクトの ARN 形式を示し、これは、Outpost が属するリージョンの AWS リージョン コード、AWS アカウント ID、Outpost ID、バケット名、およびオブジェクトキーを含みます。

```
arn:aws:s3-outposts:us-west-2:123456789012:outpost/ op-01ac5d28a6a232904/
bucket/example-s3-bucket1/object/myobject
```

オブジェクトキーの詳細については、「S3 on Outposts オブジェクトの操作」を参照してください。

S3 バージョニング

Outposts バケットの S3 バージョニングを使用して、オブジェクトの複数のバリアントを同じバケットに保持できます。S3 バージョニングを使用すると、 バケットに保存されたあらゆるオブジェクトのあらゆるバージョンを保存、取得、復元することができます。S3 バージョニングによって、意図しないユーザーアクションやアプリケーション障害から復旧できます。

詳細については、「S3 on Outposts バケットの S3 バージョニングの管理」を参照してください。

バージョン ID

バケットで S3 バージョニングを有効にすると、S3 on Outposts はバケットに追加されたすべてのオブジェクトに一意のバージョン ID を与えます。バージョニングを有効にした時点でバケットにすでに存在していたオブジェクトのバージョン ID は `null` です。これらの (またはその他の) オブジェクトを他のオペレーション (PutObject) で変更すると、新しいオブジェクトは一意のバージョン ID を取得します。

詳細については、「S3 on Outposts バケットの S3 バージョニングの管理」を参照してください。

ストレージクラスと暗号化

S3 on Outposts は、新しいストレージクラスである S3 Outposts (OUTPOSTS) を提供します。S3 Outposts のストレージクラスは、AWS Outposts のバケットに保存されたオブジェクトに対してのみ使用できます。S3 on Outposts で他の S3 ストレージクラスを使用しようとすると、S3 on Outposts は `InvalidStorageClass` エラーを返します。

デフォルトでは、S3 Outposts (OUTPOSTS) ストレージクラスに保存されたオブジェクトは、Amazon S3 マネージド暗号化キーによるサーバー側の暗号化 (SSE-S3) を使用して暗号化されます。詳細については、「S3 on Outposts のデータ暗号化」を参照してください。

バケットポリシー

バケットポリシーは、リソースベースの AWS Identity and Access Management (IAM) ポリシーを使用して、バケットとその中のオブジェクトへのアクセス許可を付与できます。バケット所有者のみ

が、ポリシーをバケットに関連付けることができます。バケットに添付された許可は、バケット所有者が所有するバケットのすべてのオブジェクトに適用されます。バケットポリシーのサイズは 20 KB に制限されています。

バケットポリシーは、AWS で標準である JSON ベースの IAM ポリシー言語を使用します。バケットポリシーを使用して、バケット内のオブジェクトに対する許可を追加または拒否できます。バケットポリシーは、ポリシーの要素に基づいて、リクエストを許可または拒否します。これらの要素には、リクエスタ、S3 on Outposts アクション、リソース、およびリクエストの側面または条件 (リクエストの作成に使用された IP アドレスなど) が含まれます。例えば、バケット所有者がアップロードされるオブジェクトの完全コントロールを持っていることを確認しつつ、S3 on Outposts バケットにオブジェクトをアップロードするクロスアカウント許可を付与するバケットポリシーを作成できます。詳細については、「Amazon S3 バケットポリシーの例」を参照してください。

バケットポリシーでは、ARN やその他の値でワイルドカード文字 (*) を使用して、オブジェクトのサブセットに対する許可を付与できます。例えば、共通のプレフィックスで始まるか、`.html` などの特定の拡張子で終わるオブジェクトのグループへのアクセスをコントロールできます。

S3 on Outposts アクセスポイント

S3 on Outposts アクセスポイントは、名前付きネットワークエンドポイントであり、そのエンドポイントを使用してデータにアクセスする方法を記述した専用のアクセスポリシーを持ちます。アクセスポイントは、S3 on Outposts の共有データセットへの大規模なデータアクセスの管理を簡素化します。アクセスポイントは、バケットにアタッチされ、それを使用して、`GetObject` や `PutObject` などの S3 オブジェクト操作を実行できます。

オブジェクト操作のためにバケットを指定するときには、アクセスポイント ARN またはアクセスポイントエイリアスを使用します。アクセスポイントエイリアスの詳細については、「S3 on Outposts アクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

アクセスポイントには、そのアクセスポイントを介して行われるすべてのリクエストに S3 on Outposts が適用する個別の許可とネットワークコントロールがあります。各アクセスポイントは、基になるバケットにアタッチされたバケットポリシーと連動して機能するカスタマイズされたアクセスポイントポリシーを適用します。

詳細については、「S3 on Outposts のバケットおよびオブジェクトにアクセスする」を参照してください。

S3 on Outposts の機能

アクセス管理

S3 on Outposts には、バケットとオブジェクトへのアクセスを監査および管理する機能があります。デフォルトでは、S3 on Outposts バケットとそれらの中のオブジェクトはプライベートです。自分が作成した S3 on Outposts リソースにのみアクセスできます。

以下の機能を使用して、特定のユースケースをサポートする詳細なリソース許可を付与したり、S3 on Outposts リソースの許可を監査したりできます。

- S3 ブロックパブリックアクセス – バケットおよびオブジェクトへのパブリックアクセスをブロックします。Outposts のバケットの場合、デフォルトでは、[パブリックアクセスをブロック] は常に有効です。
- AWS Identity and Access Management (IAM) — IAM は、AWS リソース (S3 on Outposts リソースなど) へのアクセスを安全に管理するためのウェブサービスです。IAM を使用すると、ユーザーがアクセスできる AWS のリソースを制御するアクセス許可を集中管理できます。IAM を使用して、誰を認証 (サインイン) し、誰にリソースの使用を認可する (アクセス許可を付与する) かを制御します。
- S3 on Outposts アクセスポイント - S3 on Outposts の共有データセットへのデータアクセスを管理します。アクセスポイントは、専用アクセスポリシーを持つ名前付きネットワークエンドポイントです。アクセスポイントは、バケットにアタッチされ、`GetObject` や `PutObject` などのオブジェクト操作を実行するために使用できます。
- バケットポリシー – IAM ベースのポリシー言語を使用して、S3 バケットとその中のオブジェクトに対するリソースベースの許可を設定します。
- AWS Resource Access Manager (AWS RAM) — S3 on Outposts の容量を AWS アカウント 間、組織内、または AWS Organizations の組織単位 (OU) 内で安全に共有します。

ストレージのログ記録とモニタリング

S3 on Outposts には、S3 on Outposts リソースの使用状況をモニタリングおよびコントロールするためのロギングおよびモニタリングツールが用意されています。詳細については、「モニタリングツール」を参照してください。

- S3 on Outposts 用の Amazon CloudWatch メトリクス - リソースの運用状態を追跡し、容量の可用性を把握します。

- S3 on Outposts 用 Amazon CloudWatch Events イベント — S3 on Outposts API イベントのルールを作成して、Amazon Simple Queue Service (Amazon SQS)、Amazon Simple Notification Service (Amazon SNS)、AWS Lambda など、サポートされているすべての CloudWatch Events ターゲットを通じて通知を受信します。
- AWS CloudTrailS3 on Outposts のログ — S3 on Outposts でユーザー、ロール、または AWS のサービス によって実行されたアクションを記録します。CloudTrail ログを使用すると、S3 バケットレベルおよびオブジェクトレベルのオペレーションの詳細な API 追跡が可能になります。

強力な整合性

S3 on Outposts は、すべての AWS リージョン の S3 on Outposts バケット内のオブジェクトの PUT リクエストと DELETE リクエストについて、書き込み後読み取りの強力な整合性を提供します。この動作は、新しいオブジェクトの書き込みと、既存のオブジェクトを上書きする PUT リクエストの両方に適用され、DELETE リクエストにも適用されます。さらに、S3 on Outposts オブジェクトタグとオブジェクトメタデータ (HEAD オブジェクトなど) には、強力な整合性があります。詳細については、「Amazon S3 のデータ整合性モデル」を参照してください。

関連サービス

S3 on Outputs にロードしたデータは、他の AWS のサービス で使用できます。よく使用すると思われるサービスは次のとおりです。

- Amazon Elastic Compute Cloud (Amazon EC2) – 安全でスケーラブルなコンピューティング容量を AWS クラウド で提供します。Amazon EC2 の使用により、ハードウェアに事前投資する必要がなくなり、アプリケーションをより速く開発およびデプロイできます。Amazon EC2 を使用すると、必要な数 (またはそれ以下) の仮想サーバーの起動、セキュリティおよびネットワーキングの構成、ストレージの管理ができます。
- Outposts 上の Amazon Elastic Block Store - Outposts 上で Amazon EBS ローカルスナップショットを使用して、S3 on Outposts に Outpost のボリュームのスナップショットをローカルに保存できます。
- Outposts 上の Amazon Relational Database Service (Amazon RDS) — Amazon RDS ローカルバックアップを使用して、Amazon RDS バックアップを Outpost にローカルに保存できます。
- AWS DataSync - Outposts と AWS リージョン 間のデータ転送を自動化して、転送する対象、転送するタイミング、使用するネットワーク帯域幅を選択できます。S3 on Outposts は、 と統合されていますAWS DataSync 高スループットのローカル処理を必要とするオンプレミスアプリケーションの場合、S3 on Outposts は、データ転送とネットワーク変動からのバッファを最小限に抑

えるオンプレミスのオブジェクトストレージだけでなく、Outposts と AWS リージョン 間でデータを簡単に転送する機能も提供します。

S3 on Outposts へのアクセス

S3 on Outposts は、次のいずれかの方法で使用できます。

AWS Management Console

コンソールは、S3 on Outposts と AWS リソースのウェブベースのユーザーインターフェイスです。AWS アカウント にサインアップ済みの場合は、AWS Management Console にサインインし、AWS Management Console ホームページから [S3] を選択することで、S3 on Outposts にアクセスできます。次に、左のナビゲーションペインから [Outposts buckets] (Outposts バケット) を選択します。

AWS Command Line Interface

AWS コマンドラインツールを使用して、コマンドを発行するか、システムのコマンドラインでスクリプトを作成して AWS (S3 を含む) タスクを実行します。

AWS Command Line Interface (AWS CLI) は、幅広い AWS のサービス のセットに対するコマンドを提供します。AWS CLI は、Windows、macOS、Linux でサポートされています。使用を開始するには、「AWS Command Line Interfaceユーザーガイド」を参照してください。S3 on Outposts で使用できるコマンドの詳細については、「AWS CLI コマンドリファレンス」で s3api、s3control、および s3outposts を参照してください。

AWS SDK

AWS には、さまざまなプログラミング言語およびプラットフォーム
(Java、Python、Ruby、.NET、iOS、Android など) のライブラリとサンプルコードで構成された SDK (ソフトウェア開発キット) が用意されています。AWS SDK は、S3 on Outposts と AWS へのプログラムによるアクセスを作成するのに役立ちます。S3 on Outposts は Amazon S3 と同じ SDK を使用するため、S3 on Outposts は同じ S3 API、自動化、およびツールを使用して、一貫したエクスペリエンスを提供します。

S3 on Outposts は REST サービスです。AWS SDK ライブラリを使用して S3 on Outposts にリクエストを送信できます。これは、基盤となる REST API をラップして、プログラミングタスクを簡素化します。例えば、SDK は署名の計算、リクエストの暗号化による署名、エラーの管理、リクエス

トの自動再試行などのタスクを処理します。AWS SDK のダウンロードやインストールなどの詳細については、「AWS での構築ツール」を参照してください。

S3 on Outposts の支払い

Amazon EC2 インスタンスタイプ、Amazon EBS 汎用ソリッドステートドライブ (SSD) ボリューム (gp2)、および S3 on Outposts の組み合わせを特徴とするさまざまな AWS Outposts ラック構成を購入できます。価格設定には、配送、設置、インフラストラクチャサービス保守、、およびソフトウェアパッチとアップグレードが含まれます。

詳細については、「AWS Outposts ラックの料金」を参照してください。

次のステップ

S3 on Outposts の使用の詳細については、以下のトピックを参照してください。

- Outpost の設定
- Amazon S3 on Outposts と Amazon S3 の違い
- Amazon S3 on Outposts の開始方法
- S3 on Outposts のネットワーキング
- S3 on Outposts バケットの操作
- S3 on Outposts オブジェクトの操作
- S3 on Outposts のセキュリティ
- S3 on Outposts ストレージの管理
- Amazon S3 on Outposts での開発

Outpost の設定

Amazon S3 on Outposts の使用を開始するには、施設にデプロイされた Amazon S3 容量の Outpost が必要です。Outpost と S3 容量の注文オプションの詳細については、「AWS Outposts」を参照してください。Outposts に S3 容量があるかどうかを確認するには、ListOutPostsWithS3 API コールを使用できます。仕様と、S3 on Outposts が Amazon S3 とどのように異なるかについては、「Amazon S3 on Outposts と Amazon S3 の違い」を参照してください。

詳細については、以下のトピックを参照してください。

トピック

新しい Outpost をオーダーする

S3 容量を持つ新しい Outpost を注文する必要がある場合は、AWS Outposts ラックの料金を参照して、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Elastic Block Store (Amazon EBS)、Amazon S3 の容量オプションについてご確認ください。

構成を選択したら、「AWS Outposts ユーザーガイド」の「Outpost を作成して Outpost 容量を注文する」の手順に従います。

Amazon S3 on Outposts と Amazon S3 の違い

Amazon S3 on Outposts は、オンプレミスの AWS Outposts 環境にオブジェクトストレージを提供します。S3 on Outposts を使用すると、オンプレミスアプリケーションの近くにデータを維持することによって、ローカル処理、データの常駐性、および要求の厳しいパフォーマンスニーズを満たすのに役立ちます。Amazon S3 の API と機能を使用することで、S3 on Outposts は、Outposts へのデータの保存、保護、タグ付け、レポート作成、およびアクセスコントロールを容易にし、AWS インフラストラクチャをオンプレミス施設に拡張して、一貫したハイブリッドエクスペリエンスを実現します。

S3 on Outposts の独自性については、以下のトピックを参照してください。

トピック

S3 on Outposts の仕様

- Outposts のバケットの最大サイズは 50 TB です。
- AWS アカウント ごとの Outposts のバケットの最大数は 100 です。
- Outposts のバケットには、アクセスポイントとエンドポイントを使用してのみアクセスできます。

- Outposts のバケットごとのアクセスポイントの最大数は 10 です。
- アクセスポイントのポリシーのサイズは 20 KB に制限されています。
- Outpost 所有者は、AWS Resource Access Manager を使用して AWS Organizations で組織内のアクセスを管理することができます。Outpost へのアクセスを必要とするすべてのアカウントは、AWS Organizations の所有者アカウントと同じ組織内になければなりません。
- S3 on Outposts のバケット所有者アカウントは、常にバケット内のすべてのオブジェクトの所有者です。
- バケットに対し操作ができるのは、S3 on Outpostsバケット所有者アカウントのみです。
- オブジェクトのサイズの制限は、Amazon S3 と同じです。
- S3 on Outposts に保存されるすべてのオブジェクトは、OUTPOSTS ストレージクラスに保存されます。
- デフォルトでは、OUTPOSTS ストレージクラスに保存されているすべてのオブジェクトは、Amazon S3 マネージド暗号化キー (SSE-S3) によるサーバー側の暗号化を使用して保存されます。また、ユーザーが用意した暗号化キー (SSE-C) で、サーバー側の暗号化を使用してオブジェクトを保存するように明示的に選択することもできます。
- Outpost にオブジェクトを保存する十分なスペースがない場合、API は容量不足の例外 (ICE) を返します。

S3 on Outposts でサポートされている API のオペレーション

S3 on Outposts でサポートされている API オペレーションのリストについては、「Amazon S3 on Outposts の API オペレーション」を参照してください。

S3 on Outposts でサポートされていない Amazon S3 の機能

Amazon S3 の以下の機能は、現時点では Amazon S3 on Outposts でサポートされていません。使用しようとしても、拒否されます。

- アクセスコントロールリスト (ACL)
- Cross-Origin Resource Sharing (CORS)
- S3 バッチ操作
- S3 インベントリレポート
- デフォルトのバケット暗号化の変更
- パブリックバケット

- 多要素認証 (MFA) Delete
- S3 ライフサイクルの移行 (オブジェクトの削除と不完全なマルチパートアップロードの停止を除く)
- S3 オブジェクトロックのリーガルホールド
- オブジェクトロックの保持
- AWS Key Management Service (AWS KMS) キー (SSE-KMS) によるサーバー側の暗号化
- S3 Replication Time Control (S3 RTC)
- Amazon CloudWatch のリクエストメトリクス
- メトリクスの構成
- Transfer Acceleration
- S3 イベント通知
- リクエスタ支払いバケットに、
- S3 Select
- AWS Lambda のイベント
- サーバーアクセスのログ記録
- HTTP POST リクエスト
- SOAP
- ウェブサイトへのアクセス

S3 on Outposts のネットワーク要件

- S3 on Outposts のアクセスポイントにリクエストをルーティングするには、S3 on Outposts エンドポイントを作成し設定する必要があります。S3 on Outposts のエンドポイントには、次の制限が適用されます。
  - Outposts の各仮想プライベートクラウド (VPC) には、エンドポイントが 1 つ関連付けられており、Outpost 1 つにつき最大 100 のエンドポイントを設定できます。
  - 複数のアクセスポイントを同じエンドポイントにマッピングできます。
  - エンドポイントは、次の CIDR 範囲のサブスペースに CIDR ブロックがある VPC にのみ追加できます。
    - 10.0.0.0/8
    - 172.16.0.0/12
    - 192.168.0.0/16

- Outpost へのエンドポイントは、CIDR ブロックが重複しない VPC からのみ作成できます。
- エンドポイントは、その Outposts サブネット内からのみ作成できます。
- エンドポイントの作成に使用するサブネットには、S3 on Outposts で使用するための 4 つの IP アドレスが含まれている必要があります。
- ユーザー所有の IP アドレスプール (CoIP プール) を指定する場合は、S3 on Outposts で使用する 4 つの IP アドレスが含まれている必要があります。
- VPC ごとに作成できるエンドポイントは、1 つの Outpost あたり 1 つだけです。

Amazon S3 on Outposts の開始方法

Amazon S3 on Outposts を使用すると、AWS Outposts で S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを簡単に保存および取得できます。S3 on Outposts は、新しいストレージクラス、S3 Outposts (0UTP0STS) を提供し、これは Amazon S3 API を使用し、AWS Outposts 上の複数のデバイスやサーバー間でデータを永続的かつ冗長的に保存するように設計されています。仮想プライベートクラウド (VPC) を介したアクセスポイントとエンドポイント接続を使用して、Outposts バケットと通信します。Outposts バケットでは、Amazon S3 と同じ API と機能 (アクセスポリシー、暗号化、タグ付けなど) を使用できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。

Amazon S3 on Outposts を使用すると、Amazon S3 と同様に、オブジェクトストレージ、アクセスポリシー、暗号化、タグ付けなどの Amazon S3 API と機能を AWS Outposts で利用できます。S3 on Outposts の詳細については、「Amazon S3 on Outposts とは」を参照してください。

トピック

- S3 on Outposts で IAM を設定する
- AWS Management Console を使用した開始方法
- AWS CLI および SDK for Java の使用開始

S3 on Outposts で IAM を設定する

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御するために役立つ AWS のサービスです。IAM 管理者は、誰を認証 (サインイン) し、誰に Amazon S3 on Outposts リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加費用なしで使用できる AWS のサービス です。デフォルトでは、ユーザーは S3

on Outposts リソースとオペレーションへのアクセス許可を持っていません。S3 on Outposts リソースと API オペレーションへのアクセス許可を付与するには、IAM を使用してユーザー、グループ、またはロールを作成し、アクセス許可をアタッチできます。

アクセス権限を付与するには、ユーザー、グループ、またはロールにアクセス許可を追加します。

- AWS IAM Identity Center のユーザーとグループ:

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」の手順に従ってください。
- IAM 内で、ID プロバイダーによって管理されているユーザー:

  ID フェデレーションのロールを作成します。詳細については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールの作成」を参照してください。
- IAM ユーザー:
  - ユーザーが担当できるロールを作成します。手順については、「IAM ユーザーガイド」の「IAM ユーザー用ロールの作成」を参照してください。
  - (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加する。詳細については、「IAM ユーザーガイド」の「ユーザー (コンソール) へのアクセス権限の追加」を参照してください。

S3 on Outposts では、IAM ID ベースのポリシーだけでなく、バケットポリシーとアクセスポイントポリシーの両方もサポートしています。バケットポリシーとアクセスポイントポリシーは、S3 on Outposts リソースにアタッチされた、リソースベースのポリシーです。

- バケットポリシーはバケットにアタッチされ、そのバケットとバケット内のオブジェクトへのリクエストを、ポリシー内の要素に基づいて許可または拒否します。
- 対照的に、アクセスポイントポリシーはアクセスポイントにアタッチされるもので、そのアクセスポイントへのリクエストを許可または拒否します。

アクセスポイントポリシーは、基盤 S3 on Outposts バケットにアタッチされているバケットポリシーと連動します。アプリケーションやユーザーが、S3 on Outposts アクセスポイントを介して S3 on Outposts にあるオブジェクトにアクセスするためには、アクセスポイントポリシーとバケットポリシーの両方でリクエストを許可している必要があります。

アクセスポイントポリシーに含めた制限は、そのアクセスポイントを介したリクエストにのみ適用されます。例えば、アクセスポイントがバケットにアタッチされている場合、そのバケットに直接送ら

れリクエストを、アクセスポイントポリシーを使用して許可または拒否することはできません。ただし、バケットポリシーで制限を適用することで、バケットに対して直接送られる、またはアクセスポイントを介して送られるリクエストの両方を、許可または拒否することが可能です。

IAM ポリシーまたはリソースベースのポリシーで、どの S3 on Outposts アクションを許可または拒否するかを定義します。S3 on Outposts での各アクションは、S3 on Outposts の特定の API オペレーションに対応しています。S3 on Outposts では、`s3-outposts:` 名前空間が使用されます。AWS リージョン での S3 on Outposts コントロール API へのリクエストと、Outpost 上のオブジェクト API エンドポイントへのリクエストに対しては、IAM を使用した認証が行われ、`s3-outposts:` 名前空間のプレフィックスが承認されます。S3 on Outposts を使用するには、IAM ユーザーを設定し、`s3-outposts:` IAM 名前空間に対して認可します。

詳細については、「サービス認証リファレンス」の「Amazon S3 on Outposts のアクション、リソース、条件キー」を参照してください。

> **Note**
>
> - S3 on Outposts では、アクセスコントロールリスト (ACL) はサポートされていません。
> - S3 on Outposts は、バケットの所有者がオブジェクトにアクセスしたり削除したりができなくなることを防ぐために、デフォルトでバケット所有者をオブジェクト所有者として設定します。
> - S3 on Outposts では、オブジェクトがパブリックアクセスできないようにするために、常に S3 ブロックパブリックアクセスが有効になっています。

S3 on Outposts 用の IAM の設定については、以下のトピックを参照してください。

トピック

- S3 on Outposts ポリシーのプリンシパル
- S3 on Outposts のリソース ARN
- S3 on Outposts のポリシー例
- S3 on Outposts エンドポイントの許可
- S3 on Outposts でのサービスにリンクされたロール

S3 on Outposts ポリシーのプリンシパル

リソースベースのポリシーを作成して、S3 on Outposts バケットへのアクセスを許可するには、`Principal` 要素を使用して、そのリソースでのアクションまたはオペレーションに対してリクエストが可能な、ユーザーまたはアプリケーションを指定する必要があります。S3 on Outposts ポリシーには、以下のプリンシパルのいずれかが使用できます。

- AWS アカウント
- IAM ユーザー
- IAM ロール
- アクセスを特定の IP 範囲に制限するために `Condition` 要素を使用するポリシー内でワイルドカード文字 (*) を指定することによる、すべてのプリンシパル

> **Important**
>
> ポリシーに、アクセスを特定の IP アドレス範囲に制限するための `Condition` を含めない限り、`Principal` 要素内でワイルドカード文字 (*) を使用して S3 on Outposts バケットのポリシーを記述することはできません。この制限により、S3 on Outposts バケットへのパブリックアクセスを、確実に防止することができます。例については、「S3 on Outposts のポリシー例」を参照してください。

`Principal` 要素の詳細については、「IAM ユーザーガイド」の「AWS JSON ポリシーの要素: Principal」を参照してください。

S3 on Outposts のリソース ARN

S3 on Outposts の Amazon リソースネーム (ARN) には、Outpost が属する AWS リージョン、AWS アカウント ID、およびリソース名に加えて、Outposts ID が含まれます。Outposts バケットとオブジェクトにアクセスしてアクションを実行するには、次の表に示すいずれかの ARN 形式を使用する必要があります。

ARN の *`partition`* 値は AWS リージョン グループのことを指します。それぞれの AWS アカウントは 1 つのパーティションです。サポートされているパーティションは以下のとおりです。

- `aws` – AWS リージョン
- `aws-us-gov` - AWS GovCloud (US) リージョン

S3 on Outposts の ARN 形式

Amazon S3 on Outposts の ARN	ARN 形式	例
バケット ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /bucket/*bucket_name*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /bucket/*example-s3-bucket1*
アクセスポイント ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /accesspoint/ *accesspoint_name*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /accesspoint/ *access-point-name*
オブジェクト ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /bucket/*bucket_name* /object/*object_key*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /bucket/*example-s3-bucket1 /object/myobject*
S3 on Outposts アクセスポイント オブジェクトの ARN (ポリシーで使用)	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /accesspoint/ *accesspoint_name* /object/*object_key*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /accesspoint/ *access-point-name/object/myobject*

Amazon S3 on Outposts の ARN	ARN 形式	例
S3 on Outposts の ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904*

S3 on Outposts のポリシー例

Example : AWS アカウント のプリンシパルを使用する S3 on Outposts バケットポリシー

次のバケットポリシーは、AWS アカウント プリンシパルを使用して、S3 on Outposts バケットへのアクセスを許可します。このバケットポリシーを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
   "Version":"2012-10-17",
   "Id":"ExampleBucketPolicy1",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket"
      }
   ]
}
```

Example : ワイルドカードプリンシパル (*) とアクセスを特定の IP アドレス範囲に制限する条件キーを使用する、S3 on Outposts バケットポリシー

次のバケットポリシーは、ワイルドカードプリンシパル (*) を、アクセスを特定の IP アドレス範囲に制限する 条件とともに使用しています。このバケットポリシーを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
    "Version": "2012-10-17",
    "Id": "ExampleBucketPolicy2",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": { "AWS" : "*" },
            "Action":"s3-outposts:*",
            "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket",
            "Condition" : {
                "IpAddress" : {
                    "aws:SourceIp": "192.0.2.0/24"
                },
                "NotIpAddress" : {
                    "aws:SourceIp": "198.51.100.0/24"
                }
            }
        }
    ]
}
```

S3 on Outposts エンドポイントの許可

S3 on Outposts には、S3 on Outposts のエンドポイントアクションを管理するために、IAM に独自の許可が必要です。

> **Note**
>
> - お客様所有の IP アドレスプール (CoIP プール) アクセスタイプを使用するエンドポイントの場合、次の表示に示す通り、CoIP プールから IP アドレスで機能するアクセス権限を保有する必要があります。

- AWS Resource Access Manager を使って S3 on Outposts にアクセスする共有アカウントの場合では、共有アカウントのユーザーは共有サブネット上に独自のエンドポイントを作成できません。共有アカウントのユーザーが独自のエンドポイントを管理する場合、共有アカウントは Outpost に独自のサブネットを作成する必要があります。詳細については、「the section called “S3 on Outposts の共有”」を参照してください。

S3 on Outposts エンドポイントに関連する IAM 許可

アクション	IAM アクセス許可
`CreateEndpoint`	`s3-outposts:CreateEndpoint` `ec2:CreateNetworkInterface` `ec2:DescribeNetworkInterfaces` `ec2:DescribeVpcs` `ec2:DescribeSecurityGroups` `ec2:DescribeSubnets` `ec2:CreateTags` `iam:CreateServiceLinkedRole` オンプレミスのお客様所有の IP アドレスプール (CoIP プール) アクセスタイプを使用しているエンドポイントの場合、次の追加のアクセス許可が必要です。 `s3-outposts:CreateEndpoint` `ec2:DescribeCoipPools` `ec2:GetCoipPoolUsage` `ec2:AllocateAddress` `ec2:AssociateAddress`

アクション	IAM アクセス許可
	ec2:DescribeAddresses ec2:DescribeLocalGatewayRouteTableVpcAssociations
DeleteEndpoint	s3-outposts:DeleteEndpoint ec2:DeleteNetworkInterface ec2:DescribeNetworkInterfaces オンプレミスのお客様所有の IP アドレスプール (CoIP プール) アクセスタイプを使用しているエンドポイントの場合、次の追加のアクセス許可が必要です。 s3-outposts:DeleteEndpoint ec2:DisassociateAddress ec2:DescribeAddresses ec2:ReleaseAddress
ListEndpoints	s3-outposts:ListEndpoints

> **Note**
>
> IAM ポリシーでリソースタグを使用すると、アクセス許可を管理できます。

S3 on Outposts でのサービスにリンクされたロール

S3 on Outposts は IAM サービスにリンクされたロールを使用して、ユーザーに代わっていくつかのネットワークリソースを作成します。詳細については、「Amazon S3 on Outposts でのサービスにリンクされたロールの使用」を参照してください。

AWS Management Console を使用した開始方法

Amazon S3 on Outposts を使用すると、AWS Outposts で S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを簡単に保存および取得できます。S3 on Outposts は、新しいストレージクラス、S3 Outposts (OUTPOSTS) を提供し、これは Amazon S3 API を使用し、AWS Outposts 上の複数のデバイスやサーバー間でデータを永続的かつ冗長的に保存するように設計されています。仮想プライベートクラウド (VPC) を介したアクセスポイントとエンドポイント接続を使用して、Outposts バケットと通信します。Outposts バケットでは、Amazon S3 と同じ API と機能 (アクセスポリシー、暗号化、タグ付けなど) を使用できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。詳細については、「Amazon S3 on Outposts とは」を参照してください。

コンソールを使用して S3 on Outposts の使用を開始するには、以下のトピックを参照してください。AWS CLI または AWS SDK for Java を使って開始するには、「AWS CLI および SDK for Java の使用開始」を参照してください。

トピック

バケット、アクセスポイント、およびエンドポイントを作成する

以下の手順は、S3 on Outposts で最初のバケットを作成する方法を示しています。コンソールを使用してバケットを作成するときには、バケットに関連付けられたアクセスポイントとエンドポイントも作成します。これにより、バケットへのオブジェクトの保存をすぐに開始できます。

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. [Outposts バケットの作成] を選択します。
4. [バケット名] に、バケットのドメインネームシステム (DNS) に準拠する名前を入力します。

   バケット名には次の条件があります。

   - AWS アカウント、Outpost、および Outpost が属する AWS リージョン 内で一意であること。

- 3～63 文字である必要があります。
- 大文字を含めないでください。
- 先頭の文字には小文字の英文字または数字を使用する。

バケットを作成したら、その名前を変更することはできません。バケットの命名についてさらに詳しくは、「バケットの名前付け」を参照してください。

> **Important**
>
> バケット名にアカウント番号などの機密情報を含めないでください。バケット名は、バケット内のオブジェクトを参照する URL に表示されます。

5. [Outpost] で、バケットを配置する Outpost を選択します。
6. [Bucket Versioning] (バケットのバージョニング) で、S3 on Outposts バケットの S3 バージョニングの状態を次のいずれかのオプションに設定します。

   - [Disable] (無効化) (デフォルト) — バケットはバージョニング無効のままです。
   - [Enable] (有効化) — バケット内のオブジェクトの S3 バージョニングを有効にします。バケットに追加されたすべてのオブジェクトは、一意のバージョン ID を受け取ります。

   S3 バージョニングの詳細については、S3 on Outposts バケットの S3 バージョニングの管理 を参照してください。
7. (オプション) Outposts バケットに関連付けるオプションのタグを追加します。タグを使用して、個々のプロジェクトまたはプロジェクトのグループの基準を追跡したり、コスト配分タグを使用してバケットにラベルを付けたりできます。

   デフォルトでは、Outpost バケットに保存されているすべてのオブジェクトは、Amazon S3 マネージド暗号化キー (SSE-S3) によるサーバー側の暗号化を使用して保存されます。また、ユーザーが用意した暗号化キー (SSE-C) で、サーバー側の暗号化を使用してオブジェクトを保存するように明示的に選択することもできます。暗号化タイプを変更するには、REST API、AWS Command Line Interface (AWS CLI)、または AWS SDK を使用する必要があります。
8. [Outposts access point settings] (Outposts アクセスポイントの設定) セクションで、アクセスポイント名を入力します。

   S3 on Outposts アクセスポイントは、S3 on Outposts の共有データセットへの大規模なデータアクセスの管理を簡素化します。アクセスポイントは、Outposts バケットにアタッチされた名

前付きのネットワークエンドポイントで、S3 オブジェクトのオペレーションを実行するために使用できます。詳細については、「アクセスポイント」を参照してください。

アクセスポイント名は、このリージョンおよび Outpost のアカウント内で一意であり、アクセスポイントの制約と制限 に準拠している必要があります。

9. この Amazon S3 on Outposts アクセスポイントの VPC を選択します。

   VPC がない場合は、[Create VPC] (VPC を作成) を選択します。詳細については、「Virtual Private Cloud に制限されたアクセスポイントの作成」を参照してください。

   仮想プライベートクラウド (VPC) を使用すると、定義した仮想ネットワークに AWS リソースを起動できます。仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークによく似ていますが、 のスケーラブルなインフラストラクチャを使用できるというメリットがありますAWS

10. (既存の VPC の場合はオプション) エンドポイントの [Endpoint subnet] (エンドポイントサブネット) を選択します。

    サブネットは、VPC の IP アドレスの範囲です。必要なサブネットがない場合は、[Create subnet] (サブネットの作成) を選択します。詳細については、「S3 on Outposts のネットワーキング」を参照してください。

11. (既存の VPC の場合はオプション) エンドポイントの [Endpoint security group] (エンドポイントセキュリティグループ) を選択します。

    セキュリティグループは、仮想ファイアウォールとして機能し、インバウンドトラフィックとアウトバウンドトラフィックをコントロールします。

12. (既存の VPC の場合はオプション) [Endpoint access type] (エンドポイントアクセスタイプ) を選択します。

    - プライベート — VPC と共に使用します。
    - お客様所有 IP – オンプレミスネットワーク内からユーザー所有の IP アドレスプール (CoIP プール) と共に使用します。

13. (オプション) [Outpost access point policy] (Outpost アクセスポイントポリシー) を指定します。コンソールに、アクセスポイントの Amazon リソースネーム (ARN) が自動的に表示されます。これをポリシーで使用できます。

14. [Outposts バケットの作成] を選択します。

> **Note**
>
> Outpost エンドポイントが作成され、バケットが使用できるようになるまでに最長で 5 分かかることがあります。追加のバケット設定を構成するには、[View details] (詳細の表示) を選択します。

次のステップ

Amazon S3 on Outposts では、オブジェクトデータは常に Outpost に保存されます。AWS が Outpost ラックを設置すると、データの常駐要件を満たすために、データは Outpost にローカルに保たれます。オブジェクトが Outpost を離れたり、AWS リージョン 外に出たりすることはありません。AWS Management Consoleはリージョン内でホストされるため、コンソールを使用して Outpost にオブジェクトをアップロードしたり、管理したりすることはできません。ただし REST API、AWS Command Line Interface (AWS CLI)、および AWS SDK を使用して、アクセスポイントを介してオブジェクトのアップロードと管理を行うことができます。

S3 on Outposts バケット、アクセスポイント、およびエンドポイントを作成した後、AWS CLI または SDK for Java を使用して、オブジェクトをバケットにアップロードできます。詳細については、「S3 on Outposts バケットにオブジェクトをアップロードする」を参照してください。

AWS CLI および SDK for Java の使用開始

Amazon S3 on Outposts を使用すると、AWS Outposts で S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを簡単に保存および取得できます。S3 on Outposts は、新しいストレージクラス、S3 Outposts (OUTPOSTS) を提供し、これは Amazon S3 API を使用し、AWS Outposts 上の複数のデバイスやサーバー間でデータを永続的かつ冗長的に保存するように設計されています。仮想プライベートクラウド (VPC) を介したアクセスポイントとエンドポイント接続を使用して、Outposts バケットと通信します。Outposts バケットでは、Amazon S3 と同じ API と機能 (アクセスポリシー、暗号化、タグ付けなど) を使用できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。詳細については、「Amazon S3 on Outposts とは」を参照してください。

S3 on Outposts の使用を開始するには、バケット、アクセスポイント、およびエンドポイントを作成する必要があります。その後、バケットにオブジェクトをアップロードできます。以下の例は、AWS CLI および SDK for Javaを使用して、S3 on Outposts の使用を開始する方法を示していま

す。コンソールを使用して開始するには、「AWS Management Console を使用した開始方法」を参照してください。

トピック

- ステップ 1: バケットを作成する
- ステップ 2: アクセスポイントの作成
- ステップ 3: エンドポイントを作成する
- ステップ 4: S3 on Outposts バケットにオブジェクトをアップロードする

ステップ 1: バケットを作成する

以下の AWS CLI および SDK for Java の例は、S3 on Outposts バケットを作成する方法を示しています。

AWS CLI

Example

次の例では、AWS CLI を使用して S3 on Outposts バケット (`s3-outposts:CreateBucket`) を作成します。このコマンドを実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control create-bucket --bucket example-outposts-bucket --outpost-
id op-01ac5d28a6a232904
```

SDK for Java

Example

次の例では、SDK for Java を使用して S3 on Outposts バケット (`s3-outposts:CreateBucket`) を作成します。

```
import com.amazonaws.services.s3control.model.*;

public String createBucket(String bucketName) {

    CreateBucketRequest reqCreateBucket = new CreateBucketRequest()
            .withBucket(bucketName)
            .withOutpostId(OutpostId)
            .withCreateBucketConfiguration(new CreateBucketConfiguration());
```

```
    CreateBucketResult respCreateBucket =
 s3ControlClient.createBucket(reqCreateBucket);
    System.out.printf("CreateBucket Response: %s%n", respCreateBucket.toString());

    return respCreateBucket.getBucketArn();

}
```

ステップ 2: アクセスポイントの作成

Amazon S3 on Outposts バケットにアクセスするには、アクセスポイントを作成して設定する必要があります。以下の例は、AWS CLI および SDK for Java を使用して、アクセスポイントを作成する方法を示しています。

アクセスポイントは、Amazon S3 の共有データセットへの大規模なデータアクセスの管理を簡素化します。アクセスポイントは、バケットにアタッチされた名前付きのネットワークエンドポイントで、Amazon S3 オブジェクトのオペレーション (`GetObject` や `PutObject` など) を実行するために使用できます。S3 on Outposts では、Outposts バケット内の任意のオブジェクトにアクセスするにはアクセスポイントを使用する必要があります。アクセスポイントでは、仮想ホスト形式のアドレス指定のみがサポートされます。

AWS CLI

Example

次の AWS CLI の例では、Outposts バケットのアクセスポイントを作成します。このコマンドを実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control create-access-point --account-id 123456789012
 --name example-outposts-access-point --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket" --vpc-configuration VpcId=example-vpc-12345
```

SDK for Java

Example

次の SDK for Java の例では、Outposts バケットのアクセスポイントを作成します。この例を実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
import com.amazonaws.services.s3control.model.*;

public String createAccessPoint(String bucketArn, String accessPointName) {

    CreateAccessPointRequest reqCreateAP = new CreateAccessPointRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withName(accessPointName)
            .withVpcConfiguration(new VpcConfiguration().withVpcId("vpc-12345"));

    CreateAccessPointResult respCreateAP =
 s3ControlClient.createAccessPoint(reqCreateAP);
    System.out.printf("CreateAccessPoint Response: %s%n", respCreateAP.toString());

    return respCreateAP.getAccessPointArn();

}
```

ステップ 3: エンドポイントを作成する

Amazon S3 on Outposts のアクセスポイントにリクエストをルーティングするには、S3 on Outposts エンドポイントを作成し設定する必要があります。エンドポイントを作成するには、Outposts のホームリージョンへのサービスリンクとのアクティブな接続が必要です。Outpost 上の各仮想プライベートクラウド (VPC) に 1 つのエンドポイントを関連付けることができます。エンドポイントクォータの詳細については、S3 on Outposts のネットワーク要件 を参照してください。Outposts バケットにアクセスしてオブジェクトオペレーションを実行できるようにするには、エンドポイントを作成する必要があります。詳細については、「エンドポイント」を参照してください。

以下の例は、AWS CLI および SDK for Java を使用して、エンドポイントを作成する方法を示しています。エンドポイントの作成と管理に必要な許可の詳細については、「S3 on Outposts エンドポイントの許可」を参照してください。

AWS CLI

Example

次の AWS CLI の例では、VPC リソースアクセスタイプを使用して、Outpost のエンドポイントを作成します。VPC はサブネットから派生します。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1
```

次の AWS CLI の例では、アクセスタイプにユーザー所有の IP アドレスプール (CoIP プール) を使用して、Outpost のエンドポイントを作成します。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1 --access-type CustomerOwnedIp --
customer-owned-ipv4-pool ipv4pool-coip-12345678901234567
```

SDK for Java

Example

次の SDK for Java の例では、Outposts のエンドポイントを作成します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.CreateEndpointRequest;
import com.amazonaws.services.s3outposts.model.CreateEndpointResult;

public void createEndpoint() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    CreateEndpointRequest createEndpointRequest = new CreateEndpointRequest()
                .withOutpostId("op-0d79779cef3c30a40")
                .withSubnetId("subnet-8c7a57c5")
                .withSecurityGroupId("sg-ab19e0d1")
                .withAccessType("CustomerOwnedIp")
                .withCustomerOwnedIpv4Pool("ipv4pool-coip-12345678901234567");
    // Use .withAccessType and .withCustomerOwnedIpv4Pool only when the access type
 is
    // customer-owned IP address pool (CoIP pool)
    CreateEndpointResult createEndpointResult =
 s3OutpostsClient.createEndpoint(createEndpointRequest);
    System.out.println("Endpoint is created and its ARN is " +
 createEndpointResult.getEndpointArn());
}
```

ステップ 4: S3 on Outposts バケットにオブジェクトをアップロードする

オブジェクトをアップロードするには、「S3 on Outposts バケットにオブジェクトをアップロードする」を参照してください。

S3 on Outposts のネットワーキング

Amazon S3 on Outposts を使用して、ローカルのデータアクセス、データ処理、およびデータレジデンシーを必要とするアプリケーションのために、オンプレミスでオブジェクトの保存と取得ができます。このセクションでは、S3 on Outposts にアクセスするためのネットワーク要件について説明します。

トピック

- ネットワークアクセスタイプの選択
- S3 on Outposts のバケットおよびオブジェクトにアクセスする
- クロスアカウント Elastic Network Interface

ネットワークアクセスタイプの選択

Outposts の S3 には、VPC 内またはオンプレミスネットワークからアクセスできます。アクセスポイントとエンドポイント接続を使用して、Outposts バケットと通信します。この接続は、AWS ネットワーク内における VPC と S3 on Outposts バケット間のトラフィックを維持します。エンドポイントを作成するときには、エンドポイントアクセスタイプを `Private` (VPC ルーティングの場合) または `CustomerOwnedIp` (CoIP プールの場合) のどちらかに指定する必要があります。

- `Private` (VPC ルーティングの場合) - アクセスタイプを指定しなかった場合、S3 on Outposts はデフォルトで `Private` を使用します。`Private` アクセスタイプでは、VPC のインスタンスは、Outposts のリソースと通信するためにパブリック IP アドレスを必要としません。VPC 内から S3 on Outposts を操作できます。このタイプのエンドポイントは、直接的な VPC ルーティングを通じてオンプレミスネットワークからはアクセスできません。詳細については、「AWS Outposts ユーザーガイド」の「ローカルゲートウェイテーブル」を参照してください。
- `CustomerOwnedIp` (CoIP プールの場合) — `Private` アクセスタイプをデフォルト設定せず、`CustomerOwnedIp` を選択しなかった場合、IP アドレス範囲を指定する必要があります。このアクセスタイプを使用して、オンプレミスネットワークと VPC 内の両方から S3 on Outposts を操作できます。VPC 内の S3 on Outposts にアクセスする場合、トラフィックはローカルゲートウェイの帯域幅に制限されます。

S3 on Outposts のバケットおよびオブジェクトにアクセスする

S3 on Outposts バケットおよびオブジェクトにアクセスするには、次のものが必要です。

- VPC のアクセスポイント。
- 同じ VPC のエンドポイント。
- Outpost と AWS リージョン 間のアクティブな接続。Outpost をリージョンに接続する方法の詳細については、AWSOutposts ユーザーガイドのAWS リージョンへの Outpost 接続を参照してください。

S3 on Outposts 内のバケットおよびオブジェクトへのアクセスの詳細については、「S3 on Outposts バケットの操作」と「S3 on Outposts オブジェクトの操作」を参照してください。

クロスアカウント Elastic Network Interface

S3 on Outposts エンドポイントは、Amazon リソースネーム (ARN) を持つ名前付きリソースです。これらのエンドポイントが作成されると、AWS Outposts は複数のクロスアカウント Elastic Network Interface を設定します。S3 on Outposts クロスアカウント Elastic Network Interface は、他のネットワークインターフェイスと同様ですが、1 つの例外があり、S3 on Outposts は、クロスアカウント Elastic Network Interface を Amazon EC2 インスタンスに関連付けます。

S3 on Outposts のドメインネームシステム (DNS) により、クロスアカウント Elastic Network Interface を介してリクエストのロードバランシングが行われます。S3 on Outposts は、クロスアカウント Elastic Network Interface を AWS アカウント内に作成します。これは、Amazon EC2 コンソールの [Network Interface] (ネットワークインターフェイス) ペインで表示できます。

CoIP プールアクセスタイプを使用するエンドポイントの場合、S3 on Outposts は、設定された CoIP プールから IP アドレスをクロスアカウント Elastic Network Interface に割り当てて関連付けます。

S3 on Outposts バケットの操作

Amazon S3 on Outposts を使用すると、AWS Outposts に S3 バケットを作成して、ローカルデータアクセス、ローカルデータ処理、およびデータレジデンシーを必要とするアプリケーション用に、オブジェクトの保存と取得がオンプレミスで容易にできます。S3 on Outposts は、新しいストレージクラス、S3 Outposts (0UTP0STS) を提供し、これは Amazon S3 API を使用し、AWS Outposts 上の複数のデバイスやサーバー間でデータを永続的かつ冗長的に保存するように設計されていま

す。Outposts バケットでは、アクセスポリシー、暗号化、タグ付けなど、Amazon S3 と同じ API と機能を使用できます。詳細については、「Amazon S3 on Outposts とは」を参照してください。

仮想プライベートクラウド (VPC) を介したアクセスポイントおよびエンドポイント接続を使用して、Outposts バケットと通信します。S3 on Outposts バケットおよびオブジェクトにアクセスするには、VPC 用のアクセスポイントと、同じ VPC のエンドポイントが必要です。詳細については、「S3 on Outposts のネットワーキング」を参照してください。

バケット

S3 on Outposts では、バケット名は Outpost で一意であり、バケットを識別するために、Outpost が属するリージョンの AWS リージョン コード、AWS アカウント ID、Outpost ID、およびバケット名が必要です。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/bucket/bucket-name
```

詳細については、「S3 on Outposts のリソース ARN」を参照してください。

アクセスポイント

Amazon S3 on Outposts は、Outposts バケットにアクセスする唯一の手段として、Virtual Private Cloud (VPC) のみのアクセスポイントをサポートします。

アクセスポイントは、Amazon S3 の共有データセットへの大規模なデータアクセスの管理を簡素化します。アクセスポイントは、バケットにアタッチされた名前付きのネットワークエンドポイントで、Amazon S3 オブジェクトのオペレーション (`GetObject` や `PutObject` など) を実行するために使用できます。S3 on Outposts では、Outposts バケット内の任意のオブジェクトにアクセスするにはアクセスポイントを使用する必要があります。アクセスポイントでは、仮想ホスト形式のアドレス指定のみがサポートされます。

以下の例は、S3 on Outposts アクセスポイントの ARN 形式を示しています。アクセスポイント ARN には、Outpost が属するリージョンの AWS リージョン コード、AWS アカウント ID、Outpost ID、およびアクセスポイント名が含まれます。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

エンドポイント

S3 on Outposts のアクセスポイントにリクエストをルーティングするには、S3 on Outposts エンドポイントを作成し設定する必要があります。S3 on Outposts エンドポイントを使用すると、VPC を

Outposts バケットにプライベートに接続できます。Outposts エンドポイント上の S3 は、Outposts バケット上の S3 へのエントリポイントの仮想 Uniform Resource Identifier (URI) です。これらは水平にスケールされ、冗長で、可用性の高い VPC コンポーネントです。

Outposts の各仮想プライベートクラウド (VPC) には、エンドポイントが 1 つ関連付けられており、Outpost 1 つにつき最大 100 のエンドポイントを設定できます。Outposts バケットにアクセスしてオブジェクト操作を実行できるようにするには、これらのエンドポイントを作成する必要があります。これらのエンドポイントを作成することにより、S3 と S3 on Outposts で同じオペレーションを実行できるようにすることで、API モデルと動作を同じにすることもできます。

S3 on Outposts の API オペレーション

Outposts バケット API オペレーションを管理するために、S3 on Outposts は、Amazon S3 エンドポイントとは別のエンドポイントをホストします。このエンドポイントは `s3-outposts.region.amazonaws.com` です。

Amazon S3 API オペレーションを使用するには、正しい ARN 形式を使用してバケットとオブジェクトに署名する必要があります。リクエストが Amazon S3 (`s3-control.region.amazonaws.com`) または S3 on Outposts (`s3-outposts.region.amazonaws.com`) のいずれに対するものであるのかを Amazon S3 が判断できるように、API オペレーションに ARN を渡す必要があります。ARN 形式に基づき、その後、S3 はリクエストに署名し、適切にルーティングできます。

リクエストが Amazon S3 コントロールプレーンに送信されるたびに、SDK は ARN からコンポーネントを抽出し、ARN から抽出された *`outpost-id`* 値と共に追加のヘッダー `x-amz-outpost-id` を含めます。リクエストは、S3 on Outposts エンドポイントにルーティングされる前に、ARN からのサービス名を使用して署名されます。この動作は、`s3control` クライアントが扱うすべての API オペレーションに適用されます。

次の表は、Amazon S3 on Outposts の拡張 API オペレーションと、Amazon S3 に関する変更点の一覧です。

API	S3 on Outposts パラメータ値	
`CreateBucket`	ARN、Outpost ID としてのバケット名	
`ListRegionalBuckets`	Outpost ID	
`DeleteBucket`	ARN としてのバケット名	

API	S3 on Outposts パラメータ値	
DeleteBucketLifecycleConfiguration	ARN としてのバケット名	
GetBucketLifecycleConfiguration	ARN としてのバケット名	
PutBucketLifecycleConfiguration	ARN としてのバケット名	
GetBucketPolicy	ARN としてのバケット名	
PutBucketPolicy	ARN としてのバケット名	
DeleteBucketPolicy	ARN としてのバケット名	
GetBucketTagging	ARN としてのバケット名	
PutBucketTagging	ARN としてのバケット名	
DeleteBucketTagging	ARN としてのバケット名	
CreateAccessPoint	ARN としてのアクセスポイント名	
DeleteAccessPoint	ARN としてのアクセスポイント名	
GetAccessPoint	ARN としてのアクセスポイント名	
GetAccessPoint	ARN としてのアクセスポイント名	
ListAccessPoints	ARN としてのアクセスポイント名	
PutAccessPointPolicy	ARN としてのアクセスポイント名	
GetAccessPointPolicy	ARN としてのアクセスポイント名	
DeleteAccessPointPolicy	ARN としてのアクセスポイント名	

S3 on Outposts バケットを作成および管理する

S3 on Outposts バケットの作成と管理の詳細については、以下のトピックを参照してください。

トピック

S3 on Outposts バケットを作成する

Amazon S3 on Outposts を使用すると、AWS Outposts で S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを簡単に保存および取得できます。S3 on Outposts は、新しいストレージクラス、S3 Outposts (OUTPOSTS) を提供し、これは Amazon S3 API を使用し、AWS Outposts 上の複数のデバイスやサーバー間でデータを永続的かつ冗長的に保存するように設計されています。仮想プライベートクラウド (VPC) を介したアクセスポイントとエンドポイント接続を使用して、Outposts バケットと通信します。Outposts バケットでは、Amazon S3 と同じ API と機能 (アクセスポリシー、暗号化、タグ付けなど) を使用できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。詳細については、「Amazon S3 on Outposts とは」を参照してください。

> **Note**
>
> バケットを作成する AWS アカウント がそのバケットを所有し、アクションをコミットできる唯一のアカウントです。バケットには、Outposts、タグ、デフォルトの暗号化、アクセスポイント設定などの設定プロパティがあります。アクセスポイント設定には、仮想プライベートクラウド (VPC)、バケット内のオブジェクトにアクセスするためのアクセスポイントポリシー、およびその他のメタデータが含まれます。詳細については、「S3 on Outposts の仕様」を参照してください。

AWS PrivateLink を使用して、仮想プライベートクラウド (VPC) のインターフェイス VPC エンドポイント経由でバケットやエンドポイントの管理アクセスを提供するバケットを作成する場合は、「S3 on Outposts の AWS PrivateLink」を参照してください。

次の例は、AWS Management Console、AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用して、S3 on Outposts バケットを作成する方法を示しています。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. [Outposts バケットの作成] を選択します。
4. [バケット名] に、バケットのドメインネームシステム (DNS) に準拠する名前を入力します。

   バケット名には次の条件があります。

   - AWS アカウント、Outpost、および Outpost が属する AWS リージョン 内で一意であること。
   - 3～63 文字である必要があります。
   - 大文字を含めないでください。
   - 先頭の文字には小文字の英文字または数字を使用する。

   バケットを作成したら、その名前を変更することはできません。バケットの命名についてさらに詳しくは、「バケットの名前付け」を参照してください。

   > **Important**
   > バケット名にアカウント番号などの機密情報を含めないでください。バケット名は、バケット内のオブジェクトを参照する URL に表示されます。

5. [Outpost] で、バケットを配置する Outpost を選択します。
6. [Bucket Versioning] (バケットのバージョニング) で、S3 on Outposts バケットの S3 バージョニングの状態を次のいずれかのオプションに設定します。

   - [Disable] (無効化) (デフォルト) — バケットはバージョニング無効のままです。

- [Enable] (有効化) — バケット内のオブジェクトの S3 バージョニングを有効にします。バケットに追加されたすべてのオブジェクトは、一意のバージョン ID を受け取ります。

S3 バージョニングの詳細については、S3 on Outposts バケットの S3 バージョニングの管理 を参照してください。

7. (オプション) Outposts バケットに関連付けるオプションのタグを追加します。タグを使用して、個々のプロジェクトまたはプロジェクトのグループの基準を追跡したり、コスト配分タグを使用してバケットにラベルを付けたりできます。

   デフォルトでは、Outpost バケットに保存されているすべてのオブジェクトは、Amazon S3 マネージド暗号化キー (SSE-S3) によるサーバー側の暗号化を使用して保存されます。また、ユーザーが用意した暗号化キー (SSE-C) で、サーバー側の暗号化を使用してオブジェクトを保存するように明示的に選択することもできます。暗号化タイプを変更するには、REST API、AWS Command Line Interface (AWS CLI)、または AWS SDK を使用する必要があります。

8. [Outposts access point settings] (Outposts アクセスポイントの設定) セクションで、アクセスポイント名を入力します。

   S3 on Outposts アクセスポイントは、S3 on Outposts の共有データセットへの大規模なデータアクセスの管理を簡素化します。アクセスポイントは、Outposts バケットにアタッチされた名前付きのネットワークエンドポイントで、S3 オブジェクトのオペレーションを実行するために使用できます。詳細については、「アクセスポイント」を参照してください。

   アクセスポイント名は、このリージョンおよび Outpost のアカウント内で一意であり、アクセスポイントの制約と制限 に準拠している必要があります。

9. この Amazon S3 on Outposts アクセスポイントの VPC を選択します。

   VPC がない場合は、[Create VPC] (VPC を作成) を選択します。詳細については、「Virtual Private Cloud に制限されたアクセスポイントの作成」を参照してください。

   仮想プライベートクラウド (VPC) を使用すると、定義した仮想ネットワークに AWS リソースを起動できます。仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークによく似ていますが、 のスケーラブルなインフラストラクチャを使用できるというメリットがありますAWS

10. (既存の VPC の場合はオプション) エンドポイントの [Endpoint subnet] (エンドポイントサブネット) を選択します。

サブネットは、VPC の IP アドレスの範囲です。必要なサブネットがない場合は、[Create subnet] (サブネットの作成) を選択します。詳細については、「S3 on Outposts のネットワーキング」を参照してください。

11. (既存の VPC の場合はオプション) エンドポイントの [Endpoint security group] (エンドポイントセキュリティグループ) を選択します。

    セキュリティグループは、仮想ファイアウォールとして機能し、インバウンドトラフィックとアウトバウンドトラフィックをコントロールします。

12. (既存の VPC の場合はオプション) [Endpoint access type] (エンドポイントアクセスタイプ) を選択します。

    - プライベート — VPC と共に使用します。
    - お客様所有 IP – オンプレミスネットワーク内からユーザー所有の IP アドレスプール (CoIP プール) と共に使用します。

13. (オプション) [Outpost access point policy] (Outpost アクセスポイントポリシー) を指定します。コンソールに、アクセスポイントの Amazon リソースネーム (ARN) が自動的に表示されます。これをポリシーで使用できます。

14. [Outposts バケットの作成] を選択します。

> **Note**
>
> Outpost エンドポイントが作成され、バケットが使用できるようになるまでに最長で 5 分かかることがあります。追加のバケット設定を構成するには、[View details] (詳細の表示) を選択します。

AWS CLI を使用する場合

Example

次の例では、AWS CLI を使用して S3 on Outposts バケット (`s3-outposts:CreateBucket`) を作成します。このコマンドを実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control create-bucket --bucket example-outposts-bucket --outpost-
id op-01ac5d28a6a232904
```

AWS SDK for Java の使用

Example

次の例では、SDK for Java を使用して S3 on Outposts バケット (`s3-outposts:CreateBucket`) を作成します。

```
import com.amazonaws.services.s3control.model.*;

public String createBucket(String bucketName) {

    CreateBucketRequest reqCreateBucket = new CreateBucketRequest()
            .withBucket(bucketName)
            .withOutpostId(OutpostId)
            .withCreateBucketConfiguration(new CreateBucketConfiguration());

    CreateBucketResult respCreateBucket =
 s3ControlClient.createBucket(reqCreateBucket);
    System.out.printf("CreateBucket Response: %s%n", respCreateBucket.toString());

    return respCreateBucket.getBucketArn();

}
```

S3 on Outposts バケットのタグの追加

Amazon S3 on Outposts バケットにタグを追加して、個々のプロジェクトまたはプロジェクトのグループのストレージコストおよび他の基準を追跡できます。

> **Note**
>
> バケットを作成する AWS アカウント は、バケットを所有しており、タグを変更できる唯一のアカウントです。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. タグを編集する Outposts バケットを選択します。

4. プロパティ タブを選択します。
5. タグで、[Edit] を選択します。
6. [Add new tag] (新しいタグの追加) を選択し、[Key] (キー) とオプションの [Value] (値) を入力します。

   個々のプロジェクトまたはプロジェクトグループについて、その他の基準を追跡するために、Outposts バケットに関連付けるタグを追加します。
7. [Save changes] (変更を保存) をクリックします。

AWS CLI の使用

次の AWS CLI の例では、タグを指定する JSON ドキュメント (*tagging.json*) を現在のフォルダーで使用して、S3 on Outposts バケットにタグ付け設定を適用しています。この例を実行するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。

```
aws s3control put-bucket-tagging --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --tagging file://tagging.json

tagging.json

{
   "TagSet": [
     {
       "Key": "organization",
       "Value": "marketing"
     }
   ]
}
```

次の AWS CLI の例では、コマンドラインから直接、S3 on Outposts バケットにタグ付け設定を適用しています。

```
aws s3control put-bucket-tagging --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --tagging 'TagSet=[{Key=organization,Value=marketing}]'
```

このコマンドの詳細については、「AWS CLI リファレンス」の「put-bucket-tagging」を参照してください。

バケットポリシーを使用して Amazon S3 on Outposts バケットへのアクセスを管理する

バケットポリシーは、リソースベースの AWS Identity and Access Management (IAM) ポリシーを使用して、バケットとその中のオブジェクトへのアクセス許可を付与できます。バケット所有者のみが、ポリシーをバケットに関連付けることができます。バケットに添付された許可は、バケット所有者が所有するバケットのすべてのオブジェクトに適用されます。バケットポリシーのサイズは 20 KB に制限されています。詳細については、「バケットポリシー」を参照してください。

バケットポリシーを更新して、Amazon S3 on Outposts バケットへのアクセスを管理できます。詳細については、以下のトピックを参照してください。

トピック

- Amazon S3 on Outposts バケットのバケットポリシーを追加または編集する
- Amazon S3 on Outposts バケットのバケットポリシーを表示する
- Amazon S3 on Outposts バケットのバケットポリシーを削除する
- バケットポリシーの例

Amazon S3 on Outposts バケットのバケットポリシーを追加または編集する

バケットポリシーは、リソースベースの AWS Identity and Access Management (IAM) ポリシーを使用して、バケットとその中のオブジェクトへのアクセス許可を付与できます。バケット所有者のみが、ポリシーをバケットに関連付けることができます。バケットに添付された許可は、バケット所有者が所有するバケットのすべてのオブジェクトに適用されます。バケットポリシーのサイズは 20 KB に制限されています。詳細については、「バケットポリシー」を参照してください。

以下のトピックでは、AWS Management Console、AWS Command Line Interface (AWS CLI)、または AWS SDK for Java を使用して、Amazon S3 on Outposts バケットポリシーを更新する方法を示します。

S3 コンソールの使用

バケットポリシーを作成または編集するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。

3. バケットポリシーを編集する Outposts バケットを選択します。
4. [Permissions] (許可) タブを選択します。
5. [Outposts bucket policy] (Outposts バケットポリシー) セクションで、新しいポリシーを作成または編集するには、[Edit] (編集) を選択します。

   これで、S3 on Outposts バケットポリシーを追加または編集できます。詳細については、「S3 on Outposts で IAM を設定する」を参照してください。

AWS CLI の使用

次の AWS CLI の例では、Outposts バケットにポリシーを配置します。

1. 以下のバケットポリシーを JSON ファイルに保存します。この例では、ファイル名は `policy1.json` です。*`user input placeholders`* を、ユーザー自身の情報に置き換えます。

```
{
   "Version":"2012-10-17",
   "Id":"testBucketPolicy",
   "Statement":[
      {
         "Sid":"st1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket"
      }
   ]
}
```

2. `put-bucket-policy` CLI コマンドの一部として JSON ファイルを送信します。このコマンドを実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control put-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --policy file://policy1.json
```

AWS SDK for Java の使用

次の SDK for Java の例では、Outposts バケットにポリシーを配置します。

```
import com.amazonaws.services.s3control.model.*;

public void putBucketPolicy(String bucketArn) {

    String policy = "{\"Version\":\"2012-10-17\",\"Id\":\"testBucketPolicy\",
\"Statement\":[{\"Sid\":\"st1\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"" +
 AccountId+ "\"},\"Action\":\"s3-outposts:*\",\"Resource\":\"" + bucketArn + "\"}]}";

    PutBucketPolicyRequest reqPutBucketPolicy = new PutBucketPolicyRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withPolicy(policy);

    PutBucketPolicyResult respPutBucketPolicy =
 s3ControlClient.putBucketPolicy(reqPutBucketPolicy);
    System.out.printf("PutBucketPolicy Response: %s%n",
 respPutBucketPolicy.toString());

}
```

Amazon S3 on Outposts バケットのバケットポリシーを表示する

バケットポリシーは、リソースベースの AWS Identity and Access Management (IAM) ポリシーを使用して、バケットとその中のオブジェクトへのアクセス許可を付与できます。バケット所有者のみが、ポリシーをバケットに関連付けることができます。バケットに添付された許可は、バケット所有者が所有するバケットのすべてのオブジェクトに適用されます。バケットポリシーのサイズは 20 KB に制限されています。詳細については、「バケットポリシー」を参照してください。

以下のトピックでは、AWS Management Console、AWS Command Line Interface (AWS CLI)、または AWS SDK for Java を使用して、Amazon S3 on Outposts バケットポリシーを表示する方法を示します。

S3 コンソールの使用

バケットポリシーを作成または編集するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. アクセス許可を編集する Outposts バケットを選択します。
4. [Permissions] タブを選択します。
5. [Outposts bucket policy] (Outposts バケットポリシー) セクションで、既存のバケットポリシーを確認できます。詳細については、「S3 on Outposts で IAM を設定する」を参照してください。

AWS CLI の使用

次の AWS CLI の例では、Outposts バケットのポリシーを取得します。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control get-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

AWS SDK for Java の使用

次の SDK for Java の例では、Outposts バケットのポリシーを取得します。

```
import com.amazonaws.services.s3control.model.*;

public void getBucketPolicy(String bucketArn) {

    GetBucketPolicyRequest reqGetBucketPolicy = new GetBucketPolicyRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    GetBucketPolicyResult respGetBucketPolicy =
 s3ControlClient.getBucketPolicy(reqGetBucketPolicy);
    System.out.printf("GetBucketPolicy Response: %s%n",
 respGetBucketPolicy.toString());

}
```

Amazon S3 on Outposts バケットのバケットポリシーを削除する

バケットポリシーは、リソースベースの AWS Identity and Access Management (IAM) ポリシーを使用して、バケットとその中のオブジェクトへのアクセス許可を付与できます。バケット所有者のみが、ポリシーをバケットに関連付けることができます。バケットに添付された許可は、バケット所有

者が所有するバケットのすべてのオブジェクトに適用されます。バケットポリシーのサイズは 20 KB に制限されています。詳細については、「バケットポリシー」を参照してください。

以下のトピックでは、AWS Management Consoleまたは AWS Command Line Interface (AWS CLI) を使用して、Amazon S3 on Outposts バケットポリシーを表示する方法を示します。

S3 コンソールの使用

バケットポリシーを削除するには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. アクセス許可を編集する Outposts バケットを選択します。
4. [Permissions] タブを選択します。
5. [Outposts bucket policy] (Outposts バケットポリシー) セクションで、[Delete] (削除) を選択します。
6. 削除を確定します。

AWS CLI の使用

次の例では、AWS CLI を使用して、S3 on Outposts バケット (`s3-outposts:DeleteBucket`) のバケットポリシーを削除します。このコマンドを実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control delete-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

バケットポリシーの例

S3 on Outposts バケットポリシーを使用すると、S3 on Outposts バケット内のオブジェクトへのアクセスを保護して、適切な権限を持つユーザーだけがアクセスできるようにすることができます。適切な権限を持たない認証済みユーザーが S3 on Outposts リソースにアクセスできないようにすることもできます。

このセクションでは、S3 on Outposts バケットポリシーの一般的なユースケース例を紹介します。これらのポリシーをテストするには、*`user input placeholders`* をお客様の情報 (バケット名など) と置き換えます。

オブジェクトのセットに対するアクセス許可を付与または拒否するために、Amazon リソースネーム (ARN) やその他の値でワイルドカード文字 (*) を使用できます。例えば、共通のプレフィックスで始まるか、.html などの特定の拡張子で終わるオブジェクトのグループへのアクセスをコントロールできます。

AWS Identity and Access Management (IAM) ポリシー言語については、「S3 on Outposts で IAM を設定する」を参照してください。

> **Note**
>
> Amazon S3 コンソールを使用して s3outposts のアクセス許可をテストするときには、コンソールに必要な s3outposts:createendpoint や s3outposts:listendpoints などのアクセス許可を追加で付与する必要があります。

バケットポリシーを作成するためのその他のリソース

- S3 on Outposts バケットポリシーの作成時に使用できる IAM ポリシーアクション、リソース、条件キーのリストについては、「Amazon S3 on Outposts のアクション、リソース、条件キー」を参照してください。
- S3 on Outposts ポリシーの作成に関するガイダンスについては、「Amazon S3 on Outposts バケットのバケットポリシーを追加または編集する」を参照してください。

トピック

- 特定の IP アドレスに基づく Amazon S3 on Outposts バケットへのアクセスの管理

特定の IP アドレスに基づく Amazon S3 on Outposts バケットへのアクセスの管理

バケットポリシーは、リソースベースの AWS Identity and Access Management (IAM) ポリシーを使用して、バケットとその中のオブジェクトへのアクセス許可を付与できます。バケット所有者のみが、ポリシーをバケットに関連付けることができます。バケットに添付された許可は、バケット所有者が所有するバケットのすべてのオブジェクトに適用されます。バケットポリシーのサイズは 20 KB に制限されています。詳細については、「バケットポリシー」を参照してください。

特定の IP アドレスへのアクセスの制限

以下の例では、リクエストが指定した IP アドレス範囲から発信されたものでない限り、指定したバケット内のオブジェクトに対してすべてのユーザーが S3 on Outposts のオペレーションを実行できないようにします。

> **Note**
>
> 特定の IP アドレスへのアクセスを制限する場合は、S3 on Outposts バケットにアクセスできる VPC エンドポイント、VPC ソース IP アドレス、または外部 IP アドレスも必ず指定してください。指定しないと、適切な権限が設定されていない限り、すべてのユーザーが S3 on Outposts バケット内のオブジェクトに対して s3outposts オペレーションを実行することをポリシーで拒否されている場合、バケットにアクセスできなくなる可能性があります。

このポリシーの Condition ステートメントは、許可された IP バージョン 4 (IPv4) の IP アドレスの範囲として、*192.0.2.0/24* を識別します。

Condition ブロックでは、NotIpAddress 条件と aws:SourceIp 条件キー (AWS 全体をターゲットとする条件キー) を使用します。aws:SourceIp 条件キーは、パブリック IP アドレス範囲にのみ使用できます。これらの条件キーの詳細については、「S3 on Outposts のアクション、リソース、条件キー」を参照してください。aws:SourceIp IPv4 値は標準の CIDR 表記を使用します。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素のリファレンス」を参照してください。

> **Warning**
>
> この S3 on Outposts ポリシーを使用する前に、この例の *192.0.2.0/24* IP アドレス範囲をユースケースに適した値に置き換えてください。置き換えないと、バケットにアクセスできなくなります。

```
{
    "Version": "2012-10-17",
    "Id": "S3OutpostsPolicyId1",
    "Statement": [
        {
            "Sid": "IPAllow",
```

```
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3outposts:*",
            "Resource": [
                "arn:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
accesspoint/EXAMPLE-ACCESS-POINT-NAME"
                "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/DOC-EXAMPLE-BUCKET"
            ],
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "192.0.2.0/24"
                }
            }
        }
    ]
}
```

IPv4 アドレスと IPv6 アドレスの許可

IPv6 アドレスの使用を開始する場合は、既存の IPv4 アドレス範囲に IPv6 アドレス範囲を追加して組織のすべてのポリシーを更新することをお勧めします。こうすることで、IPv6 への移行後もポリシーが引き続き機能するようになります。

以下の S3 on Outposts バケットポリシーの例は、組織の有効な IP アドレスすべてを含めるために、IPv4 アドレス範囲と IPv6 アドレス範囲を混在させる方法を示しています。このポリシーの例では、サンプル IP アドレス (*192.0.2.1* および *2001:DB8:1234:5678::1*) へのアクセスを許可したり、アドレス *203.0.113.1* および *2001:DB8:1234:5678:ABCD::1* へのアクセスを拒否したりできます。

aws:SourceIp 条件キーは、パブリック IP アドレス範囲にのみ使用できます。aws:SourceIp の IPv6 の値は、標準の CIDR 形式で指定する必要があります。IPv6 では、0 の範囲を表すために :: の使用がサポートされています (例:2001:DB8:1234:5678::/64)。詳細については、『IAM ユーザーガイド』の「IP アドレス条件演算子」を参照してください。

> **Warning**
>
> この S3 on Outposts ポリシーを使用する前に、この例の IP アドレス範囲をユースケースに適した値に置き換えます。置き換えないと、バケットにアクセスできなくなる可能性があります。

```
{
    "Id": "S3OutpostsPolicyId2",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIPmix",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3outposts:*",
            "Resource": [
                "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/DOC-EXAMPLE-BUCKET",
                    "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-
ID/bucket/DOC-EXAMPLE-BUCKET/*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "2001:DB8:1234:5678::/64"
                    ]
                },
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "203.0.113.0/24",
                        "2001:DB8:1234:5678:ABCD::/80"
                    ]
                }
            }
        }
    ]
}
```

Amazon S3 on Outposts バケットの一覧表示

Amazon S3 on Outposts を使用すると、AWS Outposts で S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを簡単に保存および取得できます。S3 on Outposts は、新しいストレージクラス、S3 Outposts (OUTPOSTS) を提供し、これは Amazon S3 API を使用し、AWS Outposts 上の複数のデバイスやサーバー間でデータを永続的かつ冗長的に保存するように設計されています。仮想プライベートクラウド (VPC) を介したアクセスポイントとエンドポイント接続を使用して、Outposts

バケットと通信します。Outposts バケットでは、Amazon S3 と同じ API と機能 (アクセスポリシー、暗号化、タグ付けなど) を使用できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。詳細については、「Amazon S3 on Outposts とは」を参照してください。

S3 on Outposts 内のバケットの操作の詳細については、S3 on Outposts バケットの操作 を参照してください。

次の例は、AWS Management Console、AWS CLI、および AWS SDK for Java を使用して、S3 on Outposts バケットのリストを返す方法を示しています。

S3 コンソールの使用

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. [Outposts buckets] (Outposts バケット) で、S3 on Outposts バケットのリストを確認します。

AWS CLI の使用

次の AWS CLI 例では、Outpost 内のバケットのリストを取得します。このコマンドを使用するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。このコマンドの詳細については、「AWS CLI リファレンス」の「list-regional-buckets」を参照してください。

```
aws s3control list-regional-buckets --account-id 123456789012 --outpost-
id op-01ac5d28a6a232904
```

AWS SDK for Java の使用

次の SDK for Java 例では、Outpost 内のバケットのリストを取得します。詳細については、「Amazon Simple Storage Service API リファレンス」の「ListRegionalBuckets」を参照してください。

```
import com.amazonaws.services.s3control.model.*;

public void listRegionalBuckets() {

    ListRegionalBucketsRequest reqListBuckets = new ListRegionalBucketsRequest()
            .withAccountId(AccountId)
            .withOutpostId(OutpostId);
```

```
    ListRegionalBucketsResult respListBuckets =
 s3ControlClient.listRegionalBuckets(reqListBuckets);
    System.out.printf("ListRegionalBuckets Response: %s%n",
 respListBuckets.toString());

}
```

AWS CLI および SDK for Java を使用して、S3 on Outposts バケットを取得する

Amazon S3 on Outposts を使用すると、AWS Outposts で S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを簡単に保存および取得できます。S3 on Outposts は、新しいストレージクラス、S3 Outposts (OUTPOSTS) を提供し、これは Amazon S3 API を使用し、AWS Outposts 上の複数のデバイスやサーバー間でデータを永続的かつ冗長的に保存するように設計されています。仮想プライベートクラウド (VPC) を介したアクセスポイントとエンドポイント接続を使用して、Outposts バケットと通信します。Outposts バケットでは、Amazon S3 と同じ API と機能 (アクセスポリシー、暗号化、タグ付けなど) を使用できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。詳細については、「Amazon S3 on Outposts とは」を参照してください。

以下の例は、AWS CLI と AWS SDK for Java を使用して、S3 on Outposts バケットを取得する方法を示しています。

> **Note**
>
> AWS CLI または AWS SDK を通じて Amazon S3 on Outposts を使用する場合、バケット名の代わりに Outposts のアクセスポイント ARN を提供します。アクセスポイント ARN は次の形式になります。*region* は Outpost が属するリージョンの AWS リージョン コードです。
>
> `arn:aws:s3-outposts:`*region*`:`*123456789012*`:outpost/`*op-01ac5d28a6a232904*`/accesspoint/`*example-outposts-access-point*
>
> S3 on Outposts ARN の詳細については、「S3 on Outposts のリソース ARN」を参照してください。

AWS CLI の使用

次の S3 on Outposts の例では、AWS CLI を使用してバケットを取得します。このコマンドを使用するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。詳細については、「AWS CLI リファレンス」の「get-bucket」を参照してください。

```
aws s3control get-bucket --account-id 123456789012 --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket"
```

AWS SDK for Java の使用

次の S3 on Outposts の例では、SDK for Java を使用してバケットを取得します。詳細については、「Amazon Simple Storage Service API リファレンス」の「GetBucket」を参照してください。

```
import com.amazonaws.services.s3control.model.*;

public void getBucket(String bucketArn) {

    GetBucketRequest reqGetBucket = new GetBucketRequest()
            .withBucket(bucketArn)
            .withAccountId(AccountId);

    GetBucketResult respGetBucket = s3ControlClient.getBucket(reqGetBucket);
    System.out.printf("GetBucket Response: %s%n", respGetBucket.toString());

}
```

Amazon S3 on Outposts バケットの削除

Amazon S3 on Outposts を使用すると、AWS Outposts で S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを簡単に保存および取得できます。S3 on Outposts は、新しいストレージクラス、S3 Outposts (0UTP0STS) を提供し、これは Amazon S3 API を使用し、AWS Outposts 上の複数のデバイスやサーバー間でデータを永続的かつ冗長的に保存するように設計されています。仮想プライベートクラウド (VPC) を介したアクセスポイントとエンドポイント接続を使用して、Outposts バケットと通信します。Outposts バケットでは、Amazon S3 と同じ API と機能 (アクセスポリシー、暗号化、タグ付けなど) を使用できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。詳細については、「Amazon S3 on Outposts とは」を参照してください。

S3 on Outposts 内のバケットの操作の詳細については、S3 on Outposts バケットの操作 を参照してください。

バケットを作成する AWS アカウント がそのバケットを所有し、バケットを削除できる唯一のアカウントです。

> **Note**
>
> - Outposts バケットは、削除する前に空にする必要があります。
>
> Amazon S3 コンソールは、S3 on Outposts オブジェクトアクションをサポートしていません。S3 on Outposts バケット内のオブジェクトを削除するには、REST API、AWS CLI、または AWS SDK を使用する必要があります。
> - Outposts バケットを削除する前に、バケットの Outposts アクセスポイントを削除する必要があります。詳細については、「アクセスポイントの削除」を参照してください。
> - バケットが削除された後は、そのバケットを復元することはできません。

次の例は、AWS Management Console と AWS Command Line Interface (AWS CLI) を使用して、S3 on Outposts バケットを削除する方法を示しています。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. 削除するバケットを選択して、[Delete] (削除) を選択します。
4. 削除を確定します。

AWS CLI の使用

次の例では、AWS CLI を使用して S3 on Outposts バケット (`s3-outposts:DeleteBucket`) を削除します。このコマンドを実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control delete-bucket --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

Amazon S3 on Outposts アクセスポイントの操作

Amazon S3 on Outposts バケットにアクセスするには、アクセスポイントを作成して設定する必要があります。

アクセスポイントは、Amazon S3 の共有データセットへの大規模なデータアクセスの管理を簡素化します。アクセスポイントは、バケットにアタッチされた名前付きのネットワークエンドポイントで、Amazon S3 オブジェクトのオペレーション (GetObject や PutObject など) を実行するために使用できます。S3 on Outposts では、Outposts バケット内の任意のオブジェクトにアクセスするにはアクセスポイントを使用する必要があります。アクセスポイントでは、仮想ホスト形式のアドレス指定のみがサポートされます。

> **Note**
>
> Outposts バケットを作成する AWS アカウント がそのバケットを所有し、そのバケットにアクセスポイントを割り当てることができる唯一のアカウントです。

以下のセクションでは、S3 on Outposts バケットを作成および管理する方法について説明します。

トピック

S3 on Outposts アクセスポイントの作成

Amazon S3 on Outposts バケットにアクセスするには、アクセスポイントを作成して設定する必要があります。

アクセスポイントは、Amazon S3 の共有データセットへの大規模なデータアクセスの管理を簡素化します。アクセスポイントは、バケットにアタッチされた名前付きのネットワークエンドポイントで、Amazon S3 オブジェクトのオペレーション (`GetObject` や `PutObject` など) を実行するために使用できます。S3 on Outposts では、Outposts バケット内の任意のオブジェクトにアクセスするにはアクセスポイントを使用する必要があります。アクセスポイントでは、仮想ホスト形式のアドレス指定のみがサポートされます。

以下の例は、AWS Management Console、AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用して、S3 on Outposts アクセスポイントを作成する方法を示しています。

> **Note**
>
> Outposts バケットを作成する AWS アカウント がそのバケットを所有し、そのバケットにアクセスポイントを割り当てることができる唯一のアカウントです。

S3 コンソールの使用

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. Outposts アクセスポイントを作成する Outposts バケットを選択します。
4. [Outposts アクセスポイント] タブを選択します。
5. [Outposts access points] (Outposts アクセスポイント) セクションで、[Create Outposts access point] (Outposts アクセスポイントの作成) を選択します。
6. [Outposts access point settings] (Outposts アクセスポイントの設定) セクションで、アクセスポイントの名前を入力し、アクセスポイントの仮想プライベートクラウド (VPC) を選択します。
7. アクセスポイントのポリシーを追加する場合は、[Outposts access point policy] (Outposts アクセスポイントポリシー) セクションに入力します。

   詳細については、「S3 on Outposts で IAM を設定する」を参照してください。

AWS CLI の使用

Example

次の AWS CLI の例では、Outposts バケットのアクセスポイントを作成します。このコマンドを実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control create-access-point --account-id 123456789012
 --name example-outposts-access-point --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket" --vpc-configuration VpcId=example-vpc-12345
```

AWS SDK for Java の使用

Example

次の SDK for Java の例では、Outposts バケットのアクセスポイントを作成します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
import com.amazonaws.services.s3control.model.*;

public String createAccessPoint(String bucketArn, String accessPointName) {

    CreateAccessPointRequest reqCreateAP = new CreateAccessPointRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withName(accessPointName)
            .withVpcConfiguration(new VpcConfiguration().withVpcId("vpc-12345"));

    CreateAccessPointResult respCreateAP =
 s3ControlClient.createAccessPoint(reqCreateAP);
    System.out.printf("CreateAccessPoint Response: %s%n", respCreateAP.toString());

    return respCreateAP.getAccessPointArn();

}
```

S3 on Outposts アクセスポイントでのバケット形式のエイリアスの使用

S3 on Outposts では、Outposts バケット内の任意のオブジェクトにアクセスするにはアクセスポイントを使用する必要があります。バケットのアクセスポイントを作成するたびに、S3 on Outposts によってアクセスポイントのエイリアスが自動的に生成されます。このアクセスポイントエイリアスは、あらゆるデータプレーンオペレーションにおいて、アクセスポイント ARN の代わりに使用できます。例えば、アクセスポイントエイリアスを使用して、PUT、GET、LIST などのオブジェクトレベルの操作を実行できます。これらのオペレーションのリストについては、オブジェクト管理のための Amazon S3 API オペレーション を参照してください。

以下は、*my-access-point* という名前のアクセスポイントの ARN とアクセスポイントのエイリアスの例です。

- アクセスポイント ARN – arn:aws:s3-outposts:*region*:*123456789012*:outpost/*op-01ac5d28a6a232904*/accesspoint/*my-access-point*
- アクセスポイントエイリアス – *my-access-po-o01ac5d28a6a232904e8xz5w8ijx1qzlbp3i3kuse10*--op-s3

ARN の詳細については、「AWS 全般のリファレンス」の「Amazon リソースネーム (ARN)」を参照してください。

アクセスポイントエイリアスの詳細については、次のトピックを参照してください。

トピック

- アクセスポイントエイリアス
- S3 on Outposts オブジェクトオペレーションでのアクセスポイントエイリアスの使用
- 制限事項

アクセスポイントエイリアス

アクセスポイントエイリアスは、S3 on Outposts バケットと同じ名前空間内に作成されます。アクセスポイントを作成すると、S3 on Outposts はアクセスポイントエイリアスを自動的に生成し、変更することはできません。アクセスポイントエイリアスは、有効な S3 on Outposts バケットの名前のすべての要件を満たしており、次の部分で構成されています。

access point name prefix-metadata--op-s3

> **Note**
>
> --op-s3 サフィックスは、アクセスポイントエイリアス用に予約されているため、バケット名やアクセスポイント名には使用しないことをお勧めします。S3 on Outposts バケット命名規則の詳細については、「S3 on Outposts バケットの操作」を参照してください。

アクセスポイントエイリアスの検索

以下の例は、Amazon S3 コンソールと AWS CLI を使用して、アクセスポイントエイリアスを検索する方法を示しています。

Example : Amazon S3 コンソールでアクセスポイントエイリアスを検索してコピーする

コンソールでアクセスポイントを作成すると、[Access Points] (アクセスポイント) リストの [Access Point alias] (アクセスポイントエイリアス) 列からアクセスポイントエイリアスを取得できます。

アクセスポイントエイリアスをコピーするには

1. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Outposts access points] (Outposts アクセスポイント) を選択します。
3. アクセスポイントエイリアスをコピーするには、次のいずれかの操作を行います。

   - [Access Points] (アクセスポイント) リストで、アクセスポイント名の横にあるオプションボタンを選択し、次に [Copy Access Point alias] (アクセスポイントエイリアスのコピー) を選択します。
   - アクセスポイント名を選択します。次に、[Outposts access point overview] (Outposts アクセスポイントの概要) で、アクセスポイントエイリアスをコピーします。

Example : AWS CLI を使用してアクセスポイントを作成し、レスポンスでのアクセスポイントエイリアスを検索する

次の `create-access-point` コマンドにおける AWS CLI の例は、アクセスポイントを作成し、自動的に生成されたアクセスポイントエイリアスを返します。このコマンドを実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control create-access-point --bucket example-outposts-bucket --name example-outposts-access-point --account-id 123456789012

{
    "AccessPointArn":
    "arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/
accesspoint/example-outposts-access-point",
    "Alias": "example-outp-o01ac5d28a6a232904e8xz5w8ijx1qzlbp3i3kuse10--op-s3"
}
```

Example : AWS CLI を使用してアクセスポイントのエイリアスを取得する

次の get-access-point コマンドにおける AWS CLI の例は、指定されたアクセスポイントに関する情報を返します。この情報には、アクセスポイントエイリアスが含まれます。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control get-access-point --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --name example-outposts-access-point --account-id 123456789012

{
    "Name": "example-outposts-access-point",
    "Bucket": "example-outposts-bucket",
    "NetworkOrigin": "Vpc",
    "VpcConfiguration": {
        "VpcId": "vpc-01234567890abcdef"
    },
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2022-09-18T17:49:15.584000+00:00",
    "Alias": "example-outp-o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3"
}
```

Example : アクセスポイントを一覧表示して、AWS CLI を使用してアクセスポイントエイリアスを検索する

次の list-access-points コマンドにおける AWS CLI の例は、指定されたアクセスポイントに関する情報を取得します。この情報には、アクセスポイントエイリアスが含まれます。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control list-access-points --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket

{
    "AccessPointList": [
        {
            "Name": "example-outposts-access-point",
```

```
            "NetworkOrigin": "Vpc",
            "VpcConfiguration": {
                "VpcId": "vpc-01234567890abcdef"
            },
            "Bucket": "example-outposts-bucket",
            "AccessPointArn": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point",
            "Alias": "example-outp-o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3"
        }
    ]
}
```

S3 on Outposts オブジェクトオペレーションでのアクセスポイントエイリアスの使用

アクセスポイントを採用する場合、大幅なコード変更を必要とせずに、アクセスポイントエイリアスを使用できます。

この AWS CLI の例は、S3 on Outposts バケットの `get-object` オペレーションを示しています。この例では、フルアクセスポイント ARN の代わりに、アクセスポイントエイリアスを `--bucket` の値として使用します。

```
aws s3api get-object --bucket my-access-po-
o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3 --key  testkey  sample-object.rtf

{
    "AcceptRanges": "bytes",
    "LastModified": "2020-01-08T22:16:28+00:00",
    "ContentLength": 910,
    "ETag": "\"00751974dc146b76404bb7290f8f51bb\"",
    "VersionId": "null",
    "ContentType": "text/rtf",
    "Metadata": {}
}
```

制限事項

- お客様はエイリアスを設定できません。
- アクセスポイントでは、エイリアスの削除、変更、無効化はできません。

- アクセスポイントエイリアスは S3 on Outposts コントロールプレーンオペレーションに使用することはできません。S3 on Outposts コントロールプレーンオペレーションのリストについては、「バケット管理のための Amazon S3 コントロール API オペレーション」を参照してください。
- エイリアスは AWS Identity and Access Management (IAM ポリシー) では使用できません。

アクセスポイント設定に関する情報を表示する

アクセスポイントは、Amazon S3 の共有データセットへの大規模なデータアクセスの管理を簡素化します。アクセスポイントは、バケットにアタッチされた名前付きのネットワークエンドポイントで、Amazon S3 オブジェクトのオペレーション (`GetObject` や `PutObject` など) を実行するために使用できます。S3 on Outposts では、Outposts バケット内の任意のオブジェクトにアクセスするにはアクセスポイントを使用する必要があります。アクセスポイントでは、仮想ホスト形式のアドレス指定のみがサポートされます。

以下のトピックでは、AWS Management Console、AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用して S3 on Outposts アクセスポイントの設定情報を返す方法について説明します。

S3 コンソールの使用

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 左のナビゲーションペインで、[Outposts access points] (Outposts アクセスポイント) を選択します。
3. 設定の詳細を表示する Outposts アクセスポイントを選択します。
4. [Outposts access point overview] (Outposts アクセスポイントの概要) で、アクセスポイント設定の詳細を確認します。

AWS CLI の使用

次の AWS CLI の例では、Outposts バケットのアクセスポイントを取得します。*user input placeholders* を、ユーザー自身の情報に置き換えます。

```
aws s3control get-access-point --account-id 123456789012 --name arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-access-point
```

AWS SDK for Java の使用

次の SDK for Java の例では、Outposts バケットのアクセスポイントを取得します。

```
import com.amazonaws.services.s3control.model.*;

public void getAccessPoint(String accessPointArn) {

    GetAccessPointRequest reqGetAP = new GetAccessPointRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn);

    GetAccessPointResult respGetAP = s3ControlClient.getAccessPoint(reqGetAP);
    System.out.printf("GetAccessPoint Response: %s%n", respGetAP.toString());

}
```

Amazon S3 on Outposts アクセスポイントのリストを表示する

アクセスポイントは、Amazon S3 の共有データセットへの大規模なデータアクセスの管理を簡素化します。アクセスポイントは、バケットにアタッチされた名前付きのネットワークエンドポイントで、Amazon S3 オブジェクトのオペレーション (`GetObject` や `PutObject` など) を実行するために使用できます。S3 on Outposts では、Outposts バケット内の任意のオブジェクトにアクセスするにはアクセスポイントを使用する必要があります。アクセスポイントでは、仮想ホスト形式のアドレス指定のみがサポートされます。

以下のトピックでは、AWS Management Console、AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用して S3 on Outposts アクセスポイントのリストを返す方法を示します。

S3 コンソールの使用

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 左のナビゲーションペインで、[Outposts access points] (Outposts アクセスポイント) を選択します。
3. [Outposts access points] (Outposts アクセスポイント) で、S3 on Outposts アクセスポイントのリストを確認します。

AWS CLI の使用

次の AWS CLI の例では、Outposts バケットのアクセスポイントを一覧表示します。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control list-access-points --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

AWS SDK for Java の使用

次の SDK for Java の例では、Outposts バケットのアクセスポイントを一覧表示します。

```
import com.amazonaws.services.s3control.model.*;

public void listAccessPoints(String bucketArn) {

    ListAccessPointsRequest reqListAPs = new ListAccessPointsRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    ListAccessPointsResult respListAPs = s3ControlClient.listAccessPoints(reqListAPs);
    System.out.printf("ListAccessPoints Response: %s%n", respListAPs.toString());

}
```

アクセスポイントの削除

アクセスポイントは、Amazon S3 の共有データセットへの大規模なデータアクセスの管理を簡素化します。アクセスポイントは、バケットにアタッチされた名前付きのネットワークエンドポイントで、Amazon S3 オブジェクトのオペレーション (GetObject や PutObject など) を実行するために使用できます。S3 on Outposts では、Outposts バケット内の任意のオブジェクトにアクセスするにはアクセスポイントを使用する必要があります。アクセスポイントでは、仮想ホスト形式のアドレス指定のみがサポートされます。

以下の例は、AWS Management Console と AWS Command Line Interface (AWS CLI) を使用して、アクセスポイントを削除する方法を示しています。

S3 コンソールの使用

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。

2. 左のナビゲーションペインで、[Outposts access points] (Outposts アクセスポイント) を選択します。
3. [Outposts access points] (Outposts アクセスポイント) セクションで、削除する Outposts アクセスポイントを選択します。
4. [削除] を選択します。
5. 削除を確定します。

AWS CLI の使用

次の AWS CLI の例では、Outposts アクセスポイントを削除します。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control delete-access-point --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

アクセスポイントポリシーを追加または編集するには

アクセスポイントには、そのアクセスポイントを介して行われるすべてのリクエストに Amazon S3 on Outposts が適用する個別の許可とネットワークコントロールがあります。各アクセスポイントは、基になるバケットにアタッチされたバケットポリシーと連動して機能するカスタマイズされたアクセスポイントポリシーを適用します。詳細については、「アクセスポイント」を参照してください。

次のトピックでは、AWS Management Console、AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用して、S3 on Outposts アクセスポイントのアクセスポイントポリシーを追加または編集する方法について説明します。

S3 コンソールの使用

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. アクセスポイントポリシーを編集する Outposts バケットを選択します。
4. [Outposts アクセスポイント] タブを選択します。
5. [Outposts access points] (Outposts アクセスポイント) セクションで、ポリシーを編集するアクセスポイントを選択し、[Edit policy] (ポリシーの編集) を選択します。

6. [Outposts access point policy] (Outposts アクセスポイントポリシー) セクションで、ポリシーを追加または編集します 。詳細については、「S3 on Outposts で IAM を設定する」を参照してください。

AWS CLI の使用

次の AWS CLI の例では、Outposts アクセスポイントにポリシーを配置します。

1. 次のアクセスポイントポリシーを JSON ファイルに保存します。この例では、ファイル名は appolicy1.json です。*user input placeholders* を、ユーザー自身の情報に置き換えます。

```
{
   "Version":"2012-10-17",
   "Id":"exampleAccessPointPolicy",
   "Statement":[
      {
         "Sid":"st1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point
      }
   ]
}
```

2. put-access-point-policy CLI コマンドの一部として JSON ファイルを送信します。*user input placeholders* を、ユーザー自身の情報に置き換えます。

```
aws s3control put-access-point-policy --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --policy file://appolicy1.json
```

AWS SDK for Java の使用

次の SDK for Java の例では、Outposts アクセスポイントにポリシーを配置します。

```
import com.amazonaws.services.s3control.model.*;

public void putAccessPointPolicy(String accessPointArn) {

    String policy = "{\"Version\":\"2012-10-17\",\"Id\":\"testAccessPointPolicy\",
\"Statement\":[{\"Sid\":\"st1\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"" +
 AccountId + "\"},\"Action\":\"s3-outposts:*\",\"Resource\":\"" + accessPointArn +
 "\"}]}";

    PutAccessPointPolicyRequest reqPutAccessPointPolicy = new
 PutAccessPointPolicyRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn)
            .withPolicy(policy);

    PutAccessPointPolicyResult respPutAccessPointPolicy =
 s3ControlClient.putAccessPointPolicy(reqPutAccessPointPolicy);
    System.out.printf("PutAccessPointPolicy Response: %s%n",
 respPutAccessPointPolicy.toString());
    printWriter.printf("PutAccessPointPolicy Response: %s%n",
 respPutAccessPointPolicy.toString());

}
```

S3 on Outposts アクセスポイントのアクセスポイントポリシーの表示

アクセスポイントには、そのアクセスポイントを介して行われるすべてのリクエストに Amazon S3 on Outposts が適用する個別の許可とネットワークコントロールがあります。各アクセスポイントは、基になるバケットにアタッチされたバケットポリシーと連動して機能するカスタマイズされたアクセスポイントポリシーを適用します。詳細については、「アクセスポイント」を参照してください。

S3 on Outposts でのアクセスポイントの操作の詳細については、「S3 on Outposts バケットの操作」を参照してください。

以下のトピックでは、AWS Management Console、AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用して、S3 on Outposts アクセスポイントポリシーを表示する方法について説明します。

S3 コンソールの使用

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。

2. 左のナビゲーションペインで、[Outposts access points] (Outposts アクセスポイント) を選択します。
3. ポリシーを表示する Outposts アクセスポイントを選択します。
4. [Permissions] (許可) タブで、S3 on Outposts アクセスポイントポリシーを確認します。
5. アクセスポイントポリシーを編集するには、「アクセスポイントポリシーを追加または編集するには」を参照してください。

AWS CLI の使用

次の AWS CLI の例では、Outposts アクセスポイントのポリシーを取得します。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control get-access-point-policy --account-id 123456789012 --name arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-access-point
```

AWS SDK for Java の使用

次の SDK for Java の例では、Outposts アクセスポイントのポリシーを取得します。

```
import com.amazonaws.services.s3control.model.*;

public void getAccessPointPolicy(String accessPointArn) {

    GetAccessPointPolicyRequest reqGetAccessPointPolicy = new
 GetAccessPointPolicyRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn);

    GetAccessPointPolicyResult respGetAccessPointPolicy =
 s3ControlClient.getAccessPointPolicy(reqGetAccessPointPolicy);
    System.out.printf("GetAccessPointPolicy Response: %s%n",
 respGetAccessPointPolicy.toString());
    printWriter.printf("GetAccessPointPolicy Response: %s%n",
 respGetAccessPointPolicy.toString());

}
```

Amazon S3 on Outposts エンドポイントの使用

Amazon S3 on Outposts のアクセスポイントにリクエストをルーティングするには、S3 on Outposts エンドポイントを作成し設定する必要があります。エンドポイントを作成するには、Outposts のホームリージョンへのサービスリンクとのアクティブな接続が必要です。Outpost 上の各仮想プライベートクラウド (VPC) に 1 つのエンドポイントを関連付けることができます。エンドポイントクォータの詳細については、S3 on Outposts のネットワーク要件 を参照してください。Outposts バケットにアクセスしてオブジェクトオペレーションを実行できるようにするには、エンドポイントを作成する必要があります。詳細については、「エンドポイント」を参照してください。

エンドポイントを作成したら、[ステータス] フィールドを使用してエンドポイントの状態を把握できます。Outposts がオフラインの場合は、CREATE_FAILED が返されます。サービスリンク接続を確認し、エンドポイントを削除して、接続が再開された後に作成オペレーションを再試行できます。その他のエラーコードのリストについては、以下を参照してください。詳細については、「エンドポイント」を参照してください。

API	ステータス	失敗理由エラーコード	メッセージ - 失敗理由
`CreateEndpoint`	Create_Failed	OutpostNotReachable	Outposts ホームリージョンへのサービスリンク接続が停止しているため、エンドポイントを作成できませんでした。接続の詳細を確認し、エンドポイントを削除して、もう一度お試しください。
`CreateEndpoint`	Create_Failed	InternalError	内部エラーのため、エンドポイントを作成できませんでした。エンドポイントを削除して、もう一度作成してください。
`DeleteEndpoint`	Delete_Failed	OutpostNotReachable	Outposts ホームリージョンへのサービスリンク接続が停止しているため、エンドポイントを削除できませんでした。接続を確認して、もう一度お試しください。

API	ステータス	失敗理由エラーコード	メッセージ - 失敗理由
DeleteEndpoint	Delete_Failed	InternalError	内部エラーのため、エンドポイントを削除できませんでした。もう一度試してください。

S3 on Outposts のバケットの操作の詳細については、S3 on Outposts バケットの操作 を参照してください。

以下のセクションでは、S3 on Outposts のエンドポイントを作成して管理する方法を説明します。

トピック

- Outpost でエンドポイントを作成する
- Amazon S3 on Outposts エンドポイントのリストを表示する
- Amazon S3 on Outposts エンドポイントの削除

Outpost でエンドポイントを作成する

Amazon S3 on Outposts のアクセスポイントにリクエストをルーティングするには、S3 on Outposts エンドポイントを作成し設定する必要があります。エンドポイントを作成するには、Outposts のホームリージョンへのサービスリンクとのアクティブな接続が必要です。Outpost 上の各仮想プライベートクラウド (VPC) に 1 つのエンドポイントを関連付けることができます。エンドポイントクォータの詳細については、S3 on Outposts のネットワーク要件 を参照してください。Outposts バケットにアクセスしてオブジェクトオペレーションを実行できるようにするには、エンドポイントを作成する必要があります。詳細については、「エンドポイント」を参照してください。

許可

エンドポイントの作成に必要な許可の詳細については、「S3 on Outposts エンドポイントの許可」を参照してください。

また、エンドポイントの作成時に、S3 on Outposts は AWS アカウントでサービスにリンクされたロールを作成します。詳細については、「Amazon S3 on Outposts でのサービスにリンクされたロールの使用」を参照してください。

以下の例は、AWS Management Console、AWS Command Line Interface (AWS CLI) および AWS SDK for Java を使用して、S3 on Outposts エンドポイントを作成する方法を示しています。

S3 コンソールの使用

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. 左のナビゲーションペインで、[Outposts access points] (Outposts アクセスポイント) を選択します。
3. [Outposts endpoints] (Outposts エンドポイント) タブを選択します。
4. [Create Outposts endpoint] (Outposts エンドポイントの作成) を選択します。
5. [Outpost] で、このエンドポイントを作成する Outpost を選択します。
6. [VPC] で、まだエンドポイントがなく、Outposts エンドポイントのルールにも準拠している VPC を選択します。

   仮想プライベートクラウド (VPC) を使用すると、定義した仮想ネットワークに AWS リソースを起動できます。仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークによく似ていますが、 のスケーラブルなインフラストラクチャを使用できるというメリットがありますAWS

   VPC がない場合は、[Create VPC] (VPC を作成) を選択します。詳細については、「Virtual Private Cloud に制限されたアクセスポイントの作成」を参照してください。
7. [Create Outposts endpoint] (Outposts エンドポイントの作成) を選択します。

AWS CLI を使用する場合

Example

次の AWS CLI の例では、VPC リソースアクセスタイプを使用して、Outpost のエンドポイントを作成します。VPC はサブネットから派生します。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1
```

次の AWS CLI の例では、アクセスタイプにユーザー所有の IP アドレスプール (CoIP プール) を使用して、Outpost のエンドポイントを作成します。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1 --access-type CustomerOwnedIp --
customer-owned-ipv4-pool ipv4pool-coip-12345678901234567
```

AWS SDK for Java の使用

Example

次の SDK for Java の例では、Outposts のエンドポイントを作成します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.CreateEndpointRequest;
import com.amazonaws.services.s3outposts.model.CreateEndpointResult;

public void createEndpoint() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    CreateEndpointRequest createEndpointRequest = new CreateEndpointRequest()
                .withOutpostId("op-0d79779cef3c30a40")
                .withSubnetId("subnet-8c7a57c5")
                .withSecurityGroupId("sg-ab19e0d1")
                .withAccessType("CustomerOwnedIp")
                .withCustomerOwnedIpv4Pool("ipv4pool-coip-12345678901234567");
    // Use .withAccessType and .withCustomerOwnedIpv4Pool only when the access type is
    // customer-owned IP address pool (CoIP pool)
    CreateEndpointResult createEndpointResult =
 s3OutpostsClient.createEndpoint(createEndpointRequest);
    System.out.println("Endpoint is created and its ARN is " +
 createEndpointResult.getEndpointArn());
}
```

Amazon S3 on Outposts エンドポイントのリストを表示する

Amazon S3 on Outposts のアクセスポイントにリクエストをルーティングするには、S3 on Outposts エンドポイントを作成し設定する必要があります。エンドポイントを作成するには、Outposts のホームリージョンへのサービスリンクとのアクティブな接続が必要です。Outpost 上の各仮想プライベートクラウド (VPC) に 1 つのエンドポイントを関連付けることができます。エンドポイントクォータの詳細については、S3 on Outposts のネットワーク要件 を参照してください。Outposts バ

ケットにアクセスしてオブジェクトオペレーションを実行できるようにするには、エンドポイントを作成する必要があります。詳細については、「エンドポイント」を参照してください。

次の例は、AWS Management Console、AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用して、S3 on Outposts エンドポイントのリストを返す方法を示しています。

S3 コンソールの使用

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 左のナビゲーションペインで、[Outposts access points] (Outposts アクセスポイント) を選択します。
3. [Outposts access points](Outposts アクセスポイント) ページで、[Outposts endpoints] (Outposts エンドポイント) タブを選択します。
4. [Outposts endpoints] (Outposts エンドポイント) で、S3 on Outposts エンドポイントのリストを表示できます。

AWS CLI を使用する場合

次の AWS CLI の例では、お客様のアカウントに関連付けられた AWS Outposts のリソースのエンドポイントを一覧表示します。このコマンドの詳細については、「AWS CLI リファレンス」の「list-endpoints」を参照してください。

```
aws s3outposts list-endpoints
```

AWS SDK for Java の使用

次のSDK for Java の例では、Outposts のエンドポイントを一覧表示します。詳細については、「Amazon Simple Storage Service API リファレンス」の「ListEndpoints」を参照してください。

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.ListEndpointsRequest;
import com.amazonaws.services.s3outposts.model.ListEndpointsResult;

public void listEndpoints() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    ListEndpointsRequest listEndpointsRequest = new ListEndpointsRequest();
```

```
    ListEndpointsResult listEndpointsResult =
 s3OutpostsClient.listEndpoints(listEndpointsRequest);
    System.out.println("List endpoints result is " + listEndpointsResult);
}
```

Amazon S3 on Outposts エンドポイントの削除

Amazon S3 on Outposts のアクセスポイントにリクエストをルーティングするには、S3 on Outposts エンドポイントを作成し設定する必要があります。エンドポイントを作成するには、Outposts のホームリージョンへのサービスリンクとのアクティブな接続が必要です。Outpost 上の各仮想プライベートクラウド (VPC) に 1 つのエンドポイントを関連付けることができます。エンドポイントクォータの詳細については、 S3 on Outposts のネットワーク要件 を参照してください。Outposts バケットにアクセスしてオブジェクトオペレーションを実行できるようにするには、エンドポイントを作成する必要があります。詳細については、「エンドポイント」を参照してください。

次の例は、AWS Management Console、AWS Command Line Interface (AWS CLI)、および AWS SDK for Java を使用して、S3 on Outposts エンドポイントを削除する方法を示しています。

S3 コンソールの使用

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 左のナビゲーションペインで、[Outposts access points] (Outposts アクセスポイント) を選択します。
3. [Outposts access points](Outposts アクセスポイント) ページで、[Outposts endpoints] (Outposts エンドポイント) タブを選択します。
4. [Outposts endpoints] (Outposts エンドポイント) で、削除するエンドポイントを選択し、[Delete] (削除) を選択します。

AWS CLI を使用する場合

次の AWS CLI の例では、Outposts のエンドポイントを削除します。このコマンドを実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3outposts delete-endpoint --endpoint-id example-endpoint-id --outpost-
id op-01ac5d28a6a232904
```

AWS SDK for Java の使用

次の SDK for Java の例では、Outposts のエンドポイントを削除します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
import com.amazonaws.arn.Arn;
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.DeleteEndpointRequest;

public void deleteEndpoint(String endpointArnInput) {
    String outpostId = "op-01ac5d28a6a232904";
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    Arn endpointArn = Arn.fromString(endpointArnInput);
    String[] resourceParts = endpointArn.getResource().getResource().split("/");
    String endpointId = resourceParts[resourceParts.length - 1];
    DeleteEndpointRequest deleteEndpointRequest = new DeleteEndpointRequest()
                .withEndpointId(endpointId)
                .withOutpostId(outpostId);
    s3OutpostsClient.deleteEndpoint(deleteEndpointRequest);
    System.out.println("Endpoint with id " + endpointId + " is deleted.");
}
```

S3 on Outposts オブジェクトの操作

Amazon S3 on Outposts を使用すると、AWS Outposts で S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを簡単に保存および取得できます。S3 on Outposts は、新しいストレージクラス、S3 Outposts (OUTPOSTS) を提供し、これは Amazon S3 API を使用し、AWS Outposts 上の複数のデバイスやサーバー間でデータを永続的かつ冗長的に保存するように設計されています。仮想プライベートクラウド (VPC) を介したアクセスポイントとエンドポイント接続を使用して、Outposts バケットと通信します。Outposts バケットでは、Amazon S3 と同じ API と機能 (アクセスポリシー、暗号化、タグ付けなど) を使用できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。

オブジェクトは、Amazon S3 on Outposts に保存される基本エンティティです。すべてのオブジェクトはバケット内に保存されます。Outpost バケット内の任意のオブジェクトにアクセスするには、アクセスポイントを使用する必要があります。オブジェクト操作のためにバケットを指定するときに

は、アクセスポイントの Amazon リソースネーム (ARN) またはアクセスポイントエイリアスを使用します。アクセスポイントエイリアスの詳細については、「S3 on Outposts アクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

次の例は、S3 on Outposts アクセスポイントの ARN 形式を示し、これは、Outpost が属するリージョンの AWS リージョン コード、AWS アカウント ID、Outpost ID、アクセスポイント名を含みます。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

S3 on Outposts ARN の詳細については、「S3 on Outposts のリソース ARN」を参照してください。

オブジェクト ARN は次の形式を使用し、Outpost が属する AWS リージョン、AWS アカウント ID、Outpost ID、バケット名、およびオブジェクトキーが含まれます。

```
arn:aws:s3-outposts:us-west-2:123456789012:outpost/ op-01ac5d28a6a232904/
bucket/example-s3-bucket1/object/myobject
```

Amazon S3 on Outposts では、オブジェクトデータは常に Outpost に保存されます。AWS が Outpost ラックを設置すると、データの常駐要件を満たすために、データは Outpost にローカルに保たれます。オブジェクトが Outpost を離れたり、AWS リージョン 外に出たりすることはありません。AWS Management Consoleはリージョン内でホストされるため、コンソールを使用して Outpost にオブジェクトをアップロードしたり、管理したりすることはできません。ただし REST API、AWS Command Line Interface (AWS CLI)、および AWS SDK を使用して、アクセスポイントを介してオブジェクトのアップロードと管理を行うことができます。

トピック

- S3 on Outposts バケットにオブジェクトをアップロードする
- AWS SDK for Java を使用して Amazon S3 on Outposts バケットにオブジェクトをコピーする
- Amazon S3 on Outposts バケットからオブジェクトを取得する
- Amazon S3 on Outposts バケット内のオブジェクトを一覧表示する
- Amazon S3 on Outposts バケット内のオブジェクトを削除する
- HeadBucket を使用して、S3 on Outposts バケットが存在し、アクセス許可があるかどうかを調べる
- SDK for Java でのマルチパートアップロードの実行と管理
- S3 on Outposts での署名付き URL の使用

- ローカルの Amazon EMR on Outposts を使用した Amazon S3 on Outposts
- 認可と認証キャッシング

S3 on Outposts バケットにオブジェクトをアップロードする

オブジェクトは、Amazon S3 on Outposts に保存される基本エンティティです。すべてのオブジェクトはバケット内に保存されます。Outpost バケット内の任意のオブジェクトにアクセスするには、アクセスポイントを使用する必要があります。オブジェクト操作のためにバケットを指定するときには、アクセスポイントの Amazon リソースネーム (ARN) またはアクセスポイントエイリアスを使用します。アクセスポイントエイリアスの詳細については、「S3 on Outposts アクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

次の例は、S3 on Outposts アクセスポイントの ARN 形式を示し、これは、Outpost が属するリージョンの AWS リージョン コード、AWS アカウント ID、Outpost ID、アクセスポイント名を含みます。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

S3 on Outposts ARN の詳細については、「S3 on Outposts のリソース ARN」を参照してください。

Amazon S3 on Outposts では、オブジェクトデータは常に Outpost に保存されます。AWS が Outpost ラックを設置すると、データの常駐要件を満たすために、データは Outpost にローカルに保たれます。オブジェクトが Outpost を離れたり、AWS リージョン 外に出たりすることはありません。AWS Management Consoleはリージョン内でホストされるため、コンソールを使用して Outpost にオブジェクトをアップロードしたり、管理したりすることはできません。ただし REST API、AWS Command Line Interface (AWS CLI)、および AWS SDK を使用して、アクセスポイントを介してオブジェクトのアップロードと管理を行うことができます。

以下の AWS CLI および AWS SDK for Java の例は、アクセスポイントを使用して S3 on Outposts バケットにオブジェクトをアップロードする方法を示しています。

AWS CLI

Example

次の例では、S3 on Outposts バケット (`s3-outposts:PutObject`) に、AWS CLI を使って `sample-object.xml` という名前のオブジェクトを配置します。このコマンドを使用するには、

それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。このコマンドの詳細については、「AWS CLI リファレンス」の「put-object」を参照してください。

```
aws s3api put-object --bucket arn:aws:s3-
outposts:Region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --key sample-object.xml --body sample-object.xml
```

SDK for Java

Example

次の例では、SDK for Java を使用して S3 on Outposts バケットにオブジェクトを配置します。この例を実行するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。詳細については、「オブジェクトのアップロード」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;

import java.io.File;

public class PutObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String stringObjKeyName = "*** String object key name ***";
        String fileObjKeyName = "*** File object key name ***";
        String fileName = "*** Path to file to upload ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Upload a text string as a new object.
            s3Client.putObject(accessPointArn, stringObjKeyName, "Uploaded String
 Object");
```

```
            // Upload a file as a new object with ContentType and title specified.
            PutObjectRequest request = new PutObjectRequest(accessPointArn,
 fileObjKeyName, new File(fileName));
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentType("plain/text");
            metadata.addUserMetadata("title", "someTitle");
            request.setMetadata(metadata);
            s3Client.putObject(request);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

AWS SDK for Java を使用して Amazon S3 on Outposts バケットにオブジェクトをコピーする

オブジェクトは、Amazon S3 on Outposts に保存される基本エンティティです。すべてのオブジェクトはバケット内に保存されます。Outpost バケット内の任意のオブジェクトにアクセスするには、アクセスポイントを使用する必要があります。オブジェクト操作のためにバケットを指定するときには、アクセスポイントの Amazon リソースネーム (ARN) またはアクセスポイントエイリアスを使用します。アクセスポイントエイリアスの詳細については、「S3 on Outposts アクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

次の例は、S3 on Outposts アクセスポイントの ARN 形式を示し、これは、Outpost が属するリージョンの AWS リージョン コード、AWS アカウント ID、Outpost ID、アクセスポイント名を含みます。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

S3 on Outposts ARN の詳細については、「S3 on Outposts のリソース ARN」を参照してください。

Amazon S3 on Outposts では、オブジェクトデータは常に Outpost に保存されます。AWS が Outpost ラックを設置すると、データの常駐要件を満たすために、データは Outpost にローカルに保たれます。オブジェクトが Outpost を離れたり、AWS リージョン 外に出たりすることはありません。AWS Management Consoleはリージョン内でホストされるため、コンソールを使用して Outpost にオブジェクトをアップロードしたり、管理したりすることはできません。ただし REST API、AWS Command Line Interface (AWS CLI)、および AWS SDK を使用して、アクセスポイントを介してオブジェクトのアップロードと管理を行うことができます。

次の例は、AWS SDK for Java を使用して、S3 on Outposts バケット内のオブジェクトをコピーする方法を示しています。

AWS SDK for Java の使用

次の S3 on Outposts の例では、SDK for Java を使用してオブジェクトを同じバケットの新しいオブジェクトにコピーします。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;

public class CopyObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String sourceKey = "*** Source object key ***";
        String destinationKey = "*** Destination object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Copy the object into a new object in the same bucket.
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(accessPointArn,
 sourceKey, accessPointArn, destinationKey);
            s3Client.copyObject(copyObjectRequest);
        } catch (AmazonServiceException e) {
```

```
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Amazon S3 on Outposts バケットからオブジェクトを取得する

オブジェクトは、Amazon S3 on Outposts に保存される基本エンティティです。すべてのオブジェクトはバケット内に保存されます。Outpost バケット内の任意のオブジェクトにアクセスするには、アクセスポイントを使用する必要があります。オブジェクト操作のためにバケットを指定するときには、アクセスポイントの Amazon リソースネーム (ARN) またはアクセスポイントエイリアスを使用します。アクセスポイントエイリアスの詳細については、「S3 on Outposts アクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

次の例は、S3 on Outposts アクセスポイントの ARN 形式を示し、これは、Outpost が属するリージョンの AWS リージョン コード、AWS アカウント ID、Outpost ID、アクセスポイント名を含みます。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

S3 on Outposts ARN の詳細については、「S3 on Outposts のリソース ARN」を参照してください。

Amazon S3 on Outposts では、オブジェクトデータは常に Outpost に保存されます。AWS が Outpost ラックを設置すると、データの常駐要件を満たすために、データは Outpost にローカルに保たれます。オブジェクトが Outpost を離れたり、AWS リージョン 外に出たりすることはありません。AWS Management Consoleはリージョン内でホストされるため、コンソールを使用して Outpost にオブジェクトをアップロードしたり、管理したりすることはできません。ただし REST API、AWS Command Line Interface (AWS CLI)、および AWS SDK を使用して、アクセスポイントを介してオブジェクトのアップロードと管理を行うことができます。

以下の例は、AWS Command Line Interface(AWS CLI) と AWS SDK for Java を使用してオブジェクトをダウンロード (取得) する方法を示しています。

AWS CLI の使用

次の例では、S3 on Outposts バケット (s3-outposts:GetObject) から、AWS CLI を使って sample-object.xml という名前のオブジェクトを取得します。このコマンドを使用するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。このコマンドの詳細については、「AWS CLI リファレンス」の「get-object」を参照してください。

```
aws s3api get-object --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point --key testkey sample-object.xml
```

AWS SDK for Java の使用

次の S3 on Outposts の例では、SDK for Java を使用してオブジェクトを取得します。この例を実行するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。詳細については、Amazon Simple Storage Service API リファレンスの GetObject を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ResponseHeaderOverrides;
import com.amazonaws.services.s3.model.S3Object;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class GetObject {
    public static void main(String[] args) throws IOException {
        String accessPointArn = "*** access point ARN ***";
        String key = "*** Object key ***";

        S3Object fullObject = null, objectPortion = null, headerOverrideObject = null;
        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
```

```
                    .build();

            // Get an object and print its contents.
            System.out.println("Downloading an object");
            fullObject = s3Client.getObject(new GetObjectRequest(accessPointArn, key));
            System.out.println("Content-Type: " +
fullObject.getObjectMetadata().getContentType());
            System.out.println("Content: ");
            displayTextInputStream(fullObject.getObjectContent());

            // Get a range of bytes from an object and print the bytes.
            GetObjectRequest rangeObjectRequest = new GetObjectRequest(accessPointArn,
key)
                    .withRange(0, 9);
            objectPortion = s3Client.getObject(rangeObjectRequest);
            System.out.println("Printing bytes retrieved.");
            displayTextInputStream(objectPortion.getObjectContent());

            // Get an entire object, overriding the specified response headers, and
print the object's content.
            ResponseHeaderOverrides headerOverrides = new ResponseHeaderOverrides()
                    .withCacheControl("No-cache")
                    .withContentDisposition("attachment; filename=example.txt");
            GetObjectRequest getObjectRequestHeaderOverride = new
GetObjectRequest(accessPointArn, key)
                    .withResponseHeaders(headerOverrides);
            headerOverrideObject = s3Client.getObject(getObjectRequestHeaderOverride);
            displayTextInputStream(headerOverrideObject.getObjectContent());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        } finally {
            // To ensure that the network connection doesn't remain open, close any
open input streams.
            if (fullObject != null) {
                fullObject.close();
            }
            if (objectPortion != null) {
                objectPortion.close();
```

```
            }
            if (headerOverrideObject != null) {
                headerOverrideObject.close();
            }
        }
    }

    private static void displayTextInputStream(InputStream input) throws IOException {
        // Read the text input stream one line at a time and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line = null;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

Amazon S3 on Outposts バケット内のオブジェクトを一覧表示する

オブジェクトは、Amazon S3 on Outposts に保存される基本エンティティです。すべてのオブジェクトはバケット内に保存されます。Outpost バケット内の任意のオブジェクトにアクセスするには、アクセスポイントを使用する必要があります。オブジェクト操作のためにバケットを指定するときには、アクセスポイントの Amazon リソースネーム (ARN) またはアクセスポイントエイリアスを使用します。アクセスポイントエイリアスの詳細については、「S3 on Outposts アクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

次の例は、S3 on Outposts アクセスポイントの ARN 形式を示し、これは、Outpost が属するリージョンの AWS リージョン コード、AWS アカウント ID、Outpost ID、アクセスポイント名を含みます。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

S3 on Outposts ARN の詳細については、「S3 on Outposts のリソース ARN」を参照してください。

> **Note**
>
> Amazon S3 on Outposts では、オブジェクトデータは常に Outpost に保存されます。AWS が Outpost ラックを設置すると、データの常駐要件を満たすために、データは Outpost に

ローカルに保たれます。オブジェクトが Outpost を離れたり、AWS リージョン 外に出たりすることはありません。AWS Management Consoleはリージョン内でホストされるため、コンソールを使用して Outpost にオブジェクトをアップロードしたり、管理したりすることはできません。ただし REST API、AWS Command Line Interface (AWS CLI)、および AWS SDK を使用して、アクセスポイントを介してオブジェクトのアップロードと管理を行うことができます。

次の例は、AWS CLI および AWS SDK for Java を使用して、S3 on Outposts バケット内のオブジェクトを一覧表示する方法を示しています。

AWS CLI の使用

次の例では、S3 on Outposts バケット (`s3-outposts:ListObjectsV2`) 内のオブジェクトを、AWS CLI を使用して一覧表示します。このコマンドを使用するには、それぞれの *`user input placeholder`* をユーザー自身の情報に置き換えます。このコマンドの詳細については、「AWS CLI リファレンス」の「list-objects-v2」を参照してください。

```
aws s3api list-objects-v2 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

> **Note**
>
> AWS SDK を通じて Amazon S3 on Outposts でこのアクションを使用する場合、バケット名の代わりに Outposts のアクセスポイント ARN を以下の形式で提供します: `arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-Outposts-Access-Point`。S3 on Outposts ARN の詳細については、「S3 on Outposts のリソース ARN」を参照してください。

AWS SDK for Java の使用

次の S3 on Outposts の例では、SDK for Java を使用してバケット内のオブジェクトを一覧表示します。この例を実行するには、それぞれの *`user input placeholder`* をユーザー自身の情報に置き換えます。

> **Important**
>
> この例では、ListObjectsAPI オペレーションの最新リビジョンである ListObjectsV2 を使用します。この改訂版の API オペレーションをアプリケーション開発に使用することをお勧めします。下位互換性のために、Amazon S3 はこの API の以前のバージョンであるオペレーションを引き続きサポートします。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsV2Request;
import com.amazonaws.services.s3.model.ListObjectsV2Result;
import com.amazonaws.services.s3.model.S3ObjectSummary;

public class ListObjectsV2 {

    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            System.out.println("Listing objects");

            // maxKeys is set to 2 to demonstrate the use of
            // ListObjectsV2Result.getNextContinuationToken()
            ListObjectsV2Request req = new
 ListObjectsV2Request().withBucketName(accessPointArn).withMaxKeys(2);
            ListObjectsV2Result result;

            do {
                result = s3Client.listObjectsV2(req);

                for (S3ObjectSummary objectSummary : result.getObjectSummaries()) {
```

```
                    System.out.printf(" - %s (size: %d)\n", objectSummary.getKey(),
 objectSummary.getSize());
                }
                // If there are more than maxKeys keys in the bucket, get a
 continuation token
                // and list the next objects.
                String token = result.getNextContinuationToken();
                System.out.println("Next Continuation Token: " + token);
                req.setContinuationToken(token);
            } while (result.isTruncated());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Amazon S3 on Outposts バケット内のオブジェクトを削除する

オブジェクトは、Amazon S3 on Outposts に保存される基本エンティティです。すべてのオブジェクトはバケット内に保存されます。Outpost バケット内の任意のオブジェクトにアクセスするには、アクセスポイントを使用する必要があります。オブジェクト操作のためにバケットを指定するときには、アクセスポイントの Amazon リソースネーム (ARN) またはアクセスポイントエイリアスを使用します。アクセスポイントエイリアスの詳細については、「S3 on Outposts アクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

次の例は、S3 on Outposts アクセスポイントの ARN 形式を示し、これは、Outpost が属するリージョンの AWS リージョン コード、AWS アカウント ID、Outpost ID、アクセスポイント名を含みます。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

S3 on Outposts ARN の詳細については、「S3 on Outposts のリソース ARN」を参照してください。

Amazon S3 on Outposts では、オブジェクトデータは常に Outpost に保存されます。AWS が Outpost ラックを設置すると、データの常駐要件を満たすために、データは Outpost にローカルに保たれます。オブジェクトが Outpost を離れたり、AWS リージョン 外に出たりすることはありません。AWS Management Consoleはリージョン内でホストされるため、コンソールを使用して Outpost にオブジェクトをアップロードしたり、管理したりすることはできません。ただし REST API、AWS Command Line Interface (AWS CLI)、および AWS SDK を使用して、アクセスポイントを介してオブジェクトのアップロードと管理を行うことができます。

次の例は、AWS Command Line Interface (AWS CLI) と AWS SDK for Java を使用して、S3 on Outposts バケット内の単一のオブジェクトまたは複数のオブジェクトを削除する方法を示しています。

AWS CLI の使用

以下の例は、S3 on Outposts バケットから単一のオブジェクトまたは複数のオブジェクトを削除する方法を示しています。

delete-object

次の例では、`sample-object.xml` を使用して、S3 on Outposts バケット (`s3-outposts:DeleteObject`) から AWS CLI という名前のオブジェクトを削除します。このコマンドを使用するには、それぞれの *`user input placeholder`* をユーザー自身の情報に置き換えます。このコマンドの詳細については、「AWS CLI リファレンス」の「delete-object」を参照してください。

```
aws s3api delete-object --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --key sample-object.xml
```

delete-objects

以下の例では、AWS CLI を使用して、S3 on Outposts バケット (`s3-outposts:DeleteObject`) から `sample-object.xml` と `test1.text` という名前の 2 つのオブジェクトを削除します。このコマンドを使用するには、それぞれの *`user input placeholder`* をユーザー自身の情報に置き換えます。このコマンドの詳細については、「AWS CLI リファレンス」の「delete-objects」を参照してください。

```
aws s3api delete-objects --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --delete file://delete.json

delete.json
{
  "Objects": [
    {
      "Key": "test1.txt"
    },
    {
      "Key": "sample-object.xml"
    }
  ],
  "Quiet": false
}
```

AWS SDK for Java の使用

以下の例は、S3 on Outposts バケットから単一のオブジェクトまたは複数のオブジェクトを削除する方法を示しています。

DeleteObject

次の S3 on Outposts の例では、SDK for Java を使用してバケット内のオブジェクトを削除します。この例を使用するには、Outpost のアクセスポイント ARN と、削除するオブジェクトのキー名を指定します。詳細については、Amazon Simple Storage Service API リファレンスの「DeleteObject」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectRequest;

public class DeleteObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String keyName = "*** key name ****";

        try {
```

```
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            s3Client.deleteObject(new DeleteObjectRequest(accessPointArn, keyName));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

DeleteObjects

次の S3 on Outposts の例では、SDK for Java を使用してバケット内のオブジェクトをアップロードして、削除します。この例を使用するには、Outpost のアクセスポイント ARN を指定します。詳細については、「Amazon Simple Storage Service API リファレンス」の「DeleteObject」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectsRequest;
import com.amazonaws.services.s3.model.DeleteObjectsRequest.KeyVersion;
import com.amazonaws.services.s3.model.DeleteObjectsResult;

import java.util.ArrayList;

public class DeleteObjects {

    public static void main(String[] args) {
       String accessPointArn = "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point";
```

```
        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Upload three sample objects.
            ArrayList<KeyVersion> keys = new ArrayList<KeyVersion>();
            for (int i = 0; i < 3; i++) {
                String keyName = "delete object example " + i;
                s3Client.putObject(accessPointArn, keyName, "Object number " + i + "
 to be deleted.");
                keys.add(new KeyVersion(keyName));
            }
            System.out.println(keys.size() + " objects successfully created.");

            // Delete the sample objects.
            DeleteObjectsRequest multiObjectDeleteRequest = new
 DeleteObjectsRequest(accessPointArn)
                    .withKeys(keys)
                    .withQuiet(false);

            // Verify that the objects were deleted successfully.
            DeleteObjectsResult delObjRes =
 s3Client.deleteObjects(multiObjectDeleteRequest);
            int successfulDeletes = delObjRes.getDeletedObjects().size();
            System.out.println(successfulDeletes + " objects successfully
 deleted.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

HeadBucket を使用して、S3 on Outposts バケットが存在し、アクセス許可があるかどうかを調べる

オブジェクトは、Amazon S3 on Outposts に保存される基本エンティティです。すべてのオブジェクトはバケット内に保存されます。Outpost バケット内の任意のオブジェクトにアクセスするには、アクセスポイントを使用する必要があります。オブジェクト操作のためにバケットを指定するときには、アクセスポイントの Amazon リソースネーム (ARN) またはアクセスポイントエイリアスを使用します。アクセスポイントエイリアスの詳細については、「S3 on Outposts アクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

次の例は、S3 on Outposts アクセスポイントの ARN 形式を示し、これは、Outpost が属するリージョンの AWS リージョン コード、AWS アカウント ID、Outpost ID、アクセスポイント名を含みます。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

S3 on Outposts ARN の詳細については、「S3 on Outposts のリソース ARN」を参照してください。

> **Note**
>
> Amazon S3 on Outposts では、オブジェクトデータは常に Outpost に保存されます。AWS が Outpost ラックを設置すると、データの常駐要件を満たすために、データは Outpost にローカルに保たれます。オブジェクトが Outpost を離れたり、AWS リージョン 外に出たりすることはありません。AWS Management Consoleはリージョン内でホストされるため、コンソールを使用して Outpost にオブジェクトをアップロードしたり、管理したりすることはできません。ただし REST API、AWS Command Line Interface (AWS CLI)、および AWS SDK を使用して、アクセスポイントを介してオブジェクトのアップロードと管理を行うことができます。

以下の AWS Command Line Interface (AWS CLI) と AWS SDK for Java の例は、HeadBucket API オペレーションを使用して、Amazon S3 on Outposts バケットが存在し、そのバケットにアクセスする許可があるかどうかを調べる方法を示しています。詳細については、「Amazon Simple Storage Service API リファレンス」の「HeadBucket」を参照してください。

AWS CLI の使用

次の S3 on Outposts AWS CLI の例では、head-bucket コマンドを使用して、バケットが存在し、そのバケットにアクセスする許可があるかどうかを調べます。このコマンドを使用するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。詳細については、「AWS CLI リファレンス」の「head-bucket」を参照してください。

```
aws s3api head-bucket --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

AWS SDK for Java の使用

次の S3 on Outposts の例は、バケットが存在するかどうか、また、そのバケットにアクセスする許可があるかどうかを調べる方法を示しています。この例を使用するには、Outpost のアクセスポイント ARN を指定します。詳細については、「Amazon Simple Storage Service API リファレンス」の「HeadBucket」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.HeadBucketRequest;

public class HeadBucket {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            s3Client.headBucket(new HeadBucketRequest(accessPointArn));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
```

```
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

SDK for Java でのマルチパートアップロードの実行と管理

Amazon S3 on Outposts を使用すると、AWS Outposts リソースで S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを保存および取得できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。詳細については、「Amazon S3 on Outposts とは」を参照してください。

以下の例は、AWS SDK for Java で S3 on Outposts を使用して、マルチパートアップロードを実行し、管理する方法を示しています。

トピック

- S3 on Outposts バケット内のオブジェクトのマルチパートアップロードを実行する
- S3 on Outposts バケット内のラージオブジェクトをマルチパートアップロードでコピーする
- S3 on Outposts バケットにオブジェクトのパーツを一覧表示する
- S3 on Outposts バケット内の進行中のマルチパートアップロードのリストを取得する

S3 on Outposts バケット内のオブジェクトのマルチパートアップロードを実行する

次の S3 on Outposts の例では、SDK for Java を使用して、バケットへオブジェクトのマルチパートアップロードを開始、アップロード、完了します。この例を実行するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。詳細については、「マルチパートアップロードを使用したオブジェクトのアップロード」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.ArrayList;
import java.util.List;
```

```
public class MultipartUploadCopy {
    public static void main(String[] args) {
        String accessPointArn = "*** Source access point ARN ***";
        String sourceObjectKey = "*** Source object key ***";
        String destObjectKey = "*** Target object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(accessPointArn, destObjectKey);
            InitiateMultipartUploadResult initResult =
 s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest(accessPointArn, sourceObjectKey);
            ObjectMetadata metadataResult =
 s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();

            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(accessPointArn)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(accessPointArn)
                        .withDestinationKey(destObjectKey)
```

```
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and make
the copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
                    accessPointArn,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
    }
```

S3 on Outposts バケット内のラージオブジェクトをマルチパートアップロードでコピーする

次の S3 on Outposts の例では、SDK for Java を使用してバケットにオブジェクトをコピーします。この例を実行するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。これは、マルチパートアップロードを使用したオブジェクトのコピー から適用した例です。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.ArrayList;
import java.util.List;

public class MultipartUploadCopy {
    public static void main(String[] args) {
        String accessPointArn = "*** Source access point ARN ***";
        String sourceObjectKey = "*** Source object key ***";
        String destObjectKey = "*** Target object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(accessPointArn, destObjectKey);
            InitiateMultipartUploadResult initResult =
 s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest(accessPointArn, sourceObjectKey);
            ObjectMetadata metadataResult =
 s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();
```

```
            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(accessPointArn)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(accessPointArn)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and make
the copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
                    accessPointArn,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
```

```
    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
    }
}
```

S3 on Outposts バケットにオブジェクトのパーツを一覧表示する

次の S3 on Outposts の例では、SDK for Java を使用してバケット内のオブジェクトのパーツを一覧表示します。この例を実行するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.List;

public class ListParts {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String keyName = "*** Key name ***";
        String uploadId = "*** Upload ID ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            ListPartsRequest listPartsRequest = new ListPartsRequest(accessPointArn,
 keyName, uploadId);
            PartListing partListing = s3Client.listParts(listPartsRequest);
            List<PartSummary> partSummaries = partListing.getParts();
```

```
            System.out.println(partSummaries.size() + " multipart upload parts");
            for (PartSummary p : partSummaries) {
                System.out.println("Upload part: Part number = \"" + p.getPartNumber()
 + "\", ETag = " + p.getETag());
            }

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

S3 on Outposts バケット内の進行中のマルチパートアップロードのリストを取得する

次の S3 on Outposts の例では、Outposts バケットから SDK for Java を使用して、進行中のマルチパートアップロードのリストを取得する方法を示します。この例を実行するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。これは、Amazon S3 用の マルチパートアップロードのリスト化 の例から適用した例です。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListMultipartUploadsRequest;
import com.amazonaws.services.s3.model.MultipartUpload;
import com.amazonaws.services.s3.model.MultipartUploadListing;

import java.util.List;

public class ListMultipartUploads {
    public static void main(String[] args) {
                String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
```

```
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Retrieve a list of all in-progress multipart uploads.
            ListMultipartUploadsRequest allMultipartUploadsRequest = new
 ListMultipartUploadsRequest(accessPointArn);
            MultipartUploadListing multipartUploadListing =
 s3Client.listMultipartUploads(allMultipartUploadsRequest);
            List<MultipartUpload> uploads =
 multipartUploadListing.getMultipartUploads();

            // Display information about all in-progress multipart uploads.
            System.out.println(uploads.size() + " multipart upload(s) in progress.");
            for (MultipartUpload u : uploads) {
                System.out.println("Upload in progress: Key = \"" + u.getKey() + "\",
 id = " + u.getUploadId());
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

S3 on Outposts での署名付き URL の使用

バケットポリシーを更新せずに、Outpost にローカルに保存されているオブジェクトへの時間制限付きのアクセス許可を付与するには、事前署名付きの URL を使用します。署名付き URL を使用すると、バケット所有者は仮想プライベートクラウド (VPC) 内の個人とオブジェクトを共有したり、オブジェクトをアップロードまたは削除する権限を、これらのユーザーに付与したりできます。

AWS SDK または AWS Command Line Interface(AWS CLI) を使用して署名付き URL を作成する際には、その URL に対し特定のアクションを関連付けます。また、1 秒から 7 日まで指定可能なカスタム有効期限を選択して、署名付き URL に対し期間制限付きのアクセス許可を付与できます。署名

付き URL を共有すると、VPC 内のユーザーは署名元のユーザーと同じように、URL に埋め込まれたアクションを実行できるようになります。この URL は有効期限が切れると失効し、以後は機能しなくなります。

署名付き URL 機能の制限

署名付き URL の機能は、それを作成したユーザーの許可によって制限されます。本質的に署名付き URL は、それらを保有しているユーザーに対しアクセスを許可するためのベアラートークンです。そのため、適切に保護することをお勧めします。

AWS Signature Version 4 (SigV4)

署名済み URL リクエストが AWS Signature Version 4 (SigV4) により認証される際に実行する特定の動作を適用するには、バケットポリシーとアクセスポイントポリシーで条件キーを使用します。例えば、`s3-outposts:signatureAge` 条件を使用するバケットポリシーを作成し、署名が作成されてから 10 分以上経過している場合に、`example-outpost-bucket` バケット内のオブジェクトに対する Amazon S3 on Outposts の署名付き URL リクエストを、すべて拒否することができます。この例を実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Deny a presigned URL request if the signature is more than 10
 minutes old",
            "Effect": "Deny",
            "Principal": {"AWS":"444455556666"},
            "Action": "s3-outposts:*",
            "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
            "Condition": {
                "NumericGreaterThan": {"s3-outposts:signatureAge": 600000},
                "StringEquals": {"s3-outposts:authType": "REST-QUERY-STRING"}
            }
        }
    ]
}
```

Signature Version 4 を使用している署名付き URL リクエストで、その認証時に実行する特定の動作を適用するための条件キー、および他のポリシー例のリストについては、「AWS Signature Version 4 (SigV4) での認証固有ポリシーキー」を参照してください。

ネットワークパスでの制限

署名付き URL の使用と、S3 on Outposts アクセスをすべて特定のネットワークパスに制限する場合は、その特定のネットワークパスを使用しながら (IAM) ポリシーを記述できます。呼び出しを行う IAM プリンシパルに対する制限を設定するには、ID ベースの AWS Identity and Access Management (IAM) ポリシー (ユーザー、グループ、ロールポリシーなど) を使用できます。S3 on Outposts リソースでの制限を設定する場合は、リソースベースのポリシー (バケットポリシーやアクセスポイントポリシーなど) を使用します。

IAM プリンシパルでのネットワークパスの制限では、これらの認証情報のユーザーは、指定したネットワークからリクエストを送信する必要があります。バケットまたはアクセスポイントの制限により、そのリソースに対するすべてのリクエストは、指定したネットワークから発信される必要があります。これらの制限は、署名付き URL のシナリオ以外でも適用されます。

使用する IAM グローバル条件は、エンドポイントのタイプによって異なります。S3 on Outposts でパブリックエンドポイントを使用している場合は、`aws:SourceIp` を使用します。S3 on Outposts の VPC エンドポイントを使用している場合は、`aws:SourceVpc` または `aws:SourceVpce` を使用します。

次の IAM ポリシーステートメントでは、プリンシパルは、指定されたネットワーク範囲からのみ AWS にアクセスする必要があります。このポリシーステートメントでは、すべてのアクセスがその範囲から発信される必要があります。これは、S3 on Outposts の署名付き URL を使用するユーザーに対しても当てはまります。この例を実行するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
{
    "Sid": "NetworkRestrictionForIAMPrincipal",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "NotIpAddressIfExists": {"aws:SourceIp": "IP-address-range"},
        "BoolIfExists": {"aws:ViaAWSService": "false"}
    }
}
```

S3 on Outposts バケットへのアクセスを特定のネットワーク範囲に制限するために、`aws:SourceIP` AWS グローバル条件キーを使用するバケットポリシーの例については、「S3 on Outposts で IAM を設定する」を参照してください。

署名付き URL を作成できるユーザー

有効なセキュリティ認証情報を持つすべてのユーザーが、署名付き URL を作成できます。しかし、VPC 内のユーザーがオブジェクトに正常にアクセスするには、署名付き URL をベースにするオペレーションの実行許可を持っているユーザーにより、この署名付き URL が作成されている必要があります。

以下の認証情報は、署名付き URL の作成に使用することができます。

- IAM インスタンスプロファイル – 最大 6 時間有効。
- AWS Security Token Service – AWS アカウント ルートユーザーや IAM ユーザーの認証情報など、永続的な認証情報を使用して署名されている場合は、最大 36 時間まで有効。
- IAM ユーザー – AWS Signature Version 4 を使用している場合は、最大 7 日間まで有効。

  最大 7 日間有効の署名付き URL を作成するには、まず、使用する SDK への IAM ユーザー認証情報 (アクセスキーとシークレットキー) の委任を行います。次に、AWS Signature Version 4 を使用して署名付き URL を生成します。

> **Note**
>
> - 一時トークンを使用して作成した署名付き URL において、URL の有効期限の終了より先にそのトークンが有効期限切れになった場合は、URL も失効します。
> - 署名付き URL は、その URL を保有しているすべてのユーザーに S3 on Outposts バケットへのアクセスを許可するため、適切な保護を行うことをお勧めします。署名付き URL の保護の詳細については、「署名付き URL 機能の制限」を参照してください。

S3 on Outposts は、どのタイミングで署名付き URL の有効期限切れの日時を確認しますか?

S3 on Outposts は、HTTP リクエスト時に署名付き URL の有効期限日時を確認します。例えば、有効期限が切れる時刻の直前にクライアントが大きなファイルのダウンロードを開始した場合は、ダウンロード中に有効期限時刻が経過しても、そのダウンロードは継続されます。しかし、接続が中断

し、クライアントがダウンロードを再開しようとした時点で有効期限切れの時刻が経過している場合には、そのダウンロードは失敗します。

署名付き URL を使用してオブジェクトを共有またはアップロードする方法の詳細については、次のトピックを参照してください。

トピック

- 署名付き URL を使用したオブジェクトの共有
- S3 on Outposts バケットにオブジェクトをアップロードするための署名付き URL の生成

署名付き URL を使用したオブジェクトの共有

バケットポリシーを更新せずに、Outpost にローカルに保存されているオブジェクトへの時間制限付きのアクセス許可を付与するには、事前署名付きの URL を使用します。署名付き URL を使用すると、バケット所有者は仮想プライベートクラウド (VPC) 内の個人とオブジェクトを共有したり、オブジェクトをアップロードまたは削除する権限を、これらのユーザーに付与したりできます。

AWS SDK または AWS Command Line Interface(AWS CLI) を使用して署名付き URL を作成する際には、その URL に対し特定のアクションを関連付けます。また、1 秒から 7 日まで指定可能なカスタム有効期限を選択して、署名付き URL に対し期間制限付きのアクセス許可を付与できます。署名付き URL を共有すると、VPC 内のユーザーは署名元のユーザーと同じように、URL に埋め込まれたアクションを実行できるようになります。この URL は有効期限が切れると失効し、以後は機能しなくなります。

署名付き URL を作成する場合には、ご自身のセキュリティ認証情報を設定し、さらに次の情報を指定する必要があります。

- Outposts バケット上の Amazon S3 アクセスポイントの Amazon リソースネーム (ARN)
- オブジェクトキー
- HTTP メソッド (オブジェクトをダウンロードするための GET)
- 有効期限の終了日時

署名付き URL は、指定した期間内でのみ有効です。つまり、有効期限が切れる日時の前に、URL で許可されているアクションを開始する必要があります。有効期限が切れる日時までは、署名付き URL を複数回使用できます。一時トークンを使用して署名付き URL を作成した場合、トークンが有効期限切れになると、URL の有効期限より前であってもその URL は失効します。

署名付き URL へのアクセス権を持つ仮想プライベートクラウド (VPC) のユーザーは、このオブジェクトにアクセスできます。例えば、プライベートのバケット内にプライベートの動画を格納している場合は、署名付き URL を生成することで、その動画を他ユーザーと共有できます。署名付き URL は、その URL を保有するすべてのユーザーに S3 on Outposts バケットへのアクセス許可を付与するため、この URL には適切な保護を行うことをお勧めします。署名付き URL の保護の詳細については、署名付き URL 機能の制限 を参照してください。

有効なセキュリティ認証情報を持つすべてのユーザーが、署名付き URL を作成できます。ただし、署名付き URL の基となるオペレーションを実行するアクセス許可を持つユーザーが、その署名付き URL を作成する必要があります。詳細については、「署名付き URL を作成できるユーザー」を参照してください。

AWS SDK とAWS CLI を使用すると、S3 on Outposts バケット内にあるオブジェクトを共有するための署名付き URL を生成できます。詳細については、以下の例を参照してください。

AWS SDK の使用

AWS SDK を使用して署名付き URL を生成します。この URL を他ユーザーに配布すると、そのユーザーがオブジェクトを取得できるようになります。

> **Note**
>
> 署名付き URL の生成に AWS SDK を使用した場合、その署名付き URL の最大有効期限は、作成時点から 7 日間となります。

Java

Example

次のサンプルコードは、署名付き URL を生成します。この URL は、Outposts バケット上の S3 からオブジェクトを取得可能にするために他のユーザーに配布できます。詳細については、「S3 on Outposts での署名付き URL の使用」を参照してください。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「使用開始」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.HttpMethod;
import com.amazonaws.SdkClientException;
```

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GeneratePresignedUrlRequest;

import java.io.IOException;
import java.net.URL;
import java.time.Instant;

public class GeneratePresignedURL {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String accessPointArn = "*** access point ARN ***";
        String objectKey = "*** object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Set the presigned URL to expire after one hour.
            java.util.Date expiration = new java.util.Date();
            long expTimeMillis = Instant.now().toEpochMilli();
            expTimeMillis += 1000 * 60 * 60;
            expiration.setTime(expTimeMillis);

            // Generate the presigned URL.
            System.out.println("Generating pre-signed URL.");
            GeneratePresignedUrlRequest generatePresignedUrlRequest =
                    new GeneratePresignedUrlRequest(accessPointArn, objectKey)
                            .withMethod(HttpMethod.GET)
                            .withExpiration(expiration);
            URL url = s3Client.generatePresignedUrl(generatePresignedUrlRequest);

            System.out.println("Pre-Signed URL: " + url.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't
 process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
```

```
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

Example

次のサンプルコードは、署名付き URL を生成します。この URL は、Outposts バケット上の S3 からオブジェクトを取得可能にするために他のユーザーに配布できます。詳細については、「S3 on Outposts での署名付き URL の使用」を参照してください。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「AWS SDK for .NET の開始方法」 を参照してください。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;

namespace Amazon.DocSamples.S3
{
    class GenPresignedURLTest
    {
        private const string accessPointArn = "*** access point ARN ***";
        private const string objectKey = "*** object key ***";
        // Specify how long the presigned URL lasts, in hours.
        private const double timeoutDuration = 12;
        // Specify your bucket Region (an example Region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            string urlString = GeneratePreSignedURL(timeoutDuration);
        }
```

```
        static string GeneratePreSignedURL(double duration)
        {
            string urlString = "";
            try
            {
                GetPreSignedUrlRequest request1 = new GetPreSignedUrlRequest
                {
                    BucketName = accessPointArn,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration)
                };
                urlString = s3Client.GetPreSignedURL(request1);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            return urlString;
        }
    }
}
```

Python

次の例では、SDK for Python (Boto3) を使用して、オブジェクトを共有するための署名付き URL を生成しています。例えば、Boto3 クライアントおよび `generate_presigned_url` 関数を使用して、オブジェクトを `GET` する署名付き URL を生成します。

```
import boto3
    url = boto3.client('s3').generate_presigned_url(
    ClientMethod='get_object',
    Params={'Bucket': 'ACCESS_POINT_ARN', 'Key': 'OBJECT_KEY'},
    ExpiresIn=3600)
```

SDK for Python (Boto3) を使用して署名付き URL を生成する方法については、「AWS SDK for Python (Boto) API リファレンス」の「Python」を参照してください。

AWS CLI の使用

次の例では、AWS CLI コマンドにより、S3 on Outposts バケットのための署名付き URL を生成します。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

> **Note**
>
> 署名付き URL の作成に AWS CLI を使用した場合、その署名付き URL の最大有効期限は、作成時点から 7 日間になります。

```
aws s3 presign s3://arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/example-outpost-access-
point/mydoc.txt --expires-in 604800
```

詳細については、「AWS CLI コマンドリファレンス」の「presign」を参照してください。

S3 on Outposts バケットにオブジェクトをアップロードするための署名付き URL の生成

バケットポリシーを更新せずに、Outpost にローカルに保存されているオブジェクトへの時間制限付きのアクセス許可を付与するには、事前署名付きの URL を使用します。署名付き URL を使用すると、バケット所有者は仮想プライベートクラウド (VPC) 内の個人とオブジェクトを共有したり、オブジェクトをアップロードまたは削除する権限を、これらのユーザーに付与したりできます。

AWS SDK または AWS Command Line Interface(AWS CLI) を使用して署名付き URL を作成する際には、その URL に対し特定のアクションを関連付けます。また、1 秒から 7 日まで指定可能なカスタム有効期限を選択して、署名付き URL に対し期間制限付きのアクセス許可を付与できます。署名付き URL を共有すると、VPC 内のユーザーは署名元のユーザーと同じように、URL に埋め込まれたアクションを実行できるようになります。この URL は有効期限が切れると失効し、以後は機能しなくなります。

署名付き URL を作成する場合には、ご自身のセキュリティ認証情報を設定し、さらに次の情報を指定する必要があります。

- Outposts バケット上の Amazon S3 アクセスポイントの Amazon リソースネーム (ARN)
- オブジェクトキー
- HTTP メソッド (オブジェクトをアップロードするための PUT)

- 有効期限の終了日時

署名付き URL は、指定した期間内でのみ有効です。つまり、有効期限が切れる日時の前に、URL で許可されているアクションを開始する必要があります。有効期限が切れる日時までは、署名付き URL を複数回使用できます。一時トークンを使用して署名付き URL を作成した場合、トークンが有効期限切れになると、URL の有効期限より前であってもその URL は失効します。

マルチパートアップロードなど、複数のステップで構成されているアクションでは、すべてのステップを有効期限が切れる前に開始する必要があります。有効期限が切れた URL を使用して、S3 on Outposts がステップの開始を試みた場合は、エラーが発生します。

署名付き URL へのアクセス権を持つ仮想プライベートクラウド (VPC) 内のユーザーは、オブジェクトをアップロードすることが可能です。例えば、署名付き URL にアクセスできる VPC 内のユーザーは、お客様のバケットへのオブジェクトのアップロードが可能です。署名付き URL は、その URL を保有するすべてのユーザーに対し、S3 on Outposts バケットへのアクセスを許可するため、これらの URL には適切な保護を行うことをお勧めします。署名付き URL の保護の詳細については、署名付き URL 機能の制限 を参照してください。

有効なセキュリティ認証情報を持つすべてのユーザーが、署名付き URL を作成できます。ただし、署名付き URL の基となるオペレーションを実行するアクセス許可を持つユーザーが、その署名付き URL を作成する必要があります。詳細については、「署名付き URL を作成できるユーザー」を参照してください。

AWS SDK を使用した S3 on Outposts オブジェクトオペレーションへの署名付き URL の生成

Java

SDK for Java 2.x

この例では、限られた時間内に S3 on Outposts バケットにオブジェクトをアップロードするために使用できる、署名付き URL を生成する方法を説明します。詳細については、「S3 on Outposts での署名付き URL の使用」を参照してください。

```
    public static void signBucket(S3Presigner presigner, String
outpostAccessPointArn, String keyName) {

        try {
            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .bucket(accessPointArn)
                    .key(keyName)
                    .contentType("text/plain")
```

```
                    .build();

            PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10))
                    .putObjectRequest(objectRequest)
                    .build();

            PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);


            String myURL = presignedRequest.url().toString();
            System.out.println("Presigned URL to upload a file to: " +myURL);
            System.out.println("Which HTTP method must be used when uploading a
file: " +
                    presignedRequest.httpRequest().method());

            // Upload content to the S3 on Outposts bucket by using this URL.
            URL url = presignedRequest.url();

            // Create the connection and use it to upload the new object by using
the presigned URL.
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestProperty("Content-Type","text/plain");
            connection.setRequestMethod("PUT");
            OutputStreamWriter out = new
OutputStreamWriter(connection.getOutputStream());
            out.write("This text was uploaded as an object by using a presigned
URL.");
            out.close();

            connection.getResponseCode();
            System.out.println("HTTP response code is " +
connection.getResponseCode());

        } catch (S3Exception e) {
            e.getStackTrace();
        } catch (IOException e) {
            e.getStackTrace();
        }
```

```
    }
```

Python

SDK for Python (Boto3)

S3 on Outposts アクションを期間限定で実行できる署名付き URL の生成方法を示します。詳細については、「S3 on Outposts での署名付き URL の使用」を参照してください。この URL でリクエストを行うには、Requests パッケージを使用します。

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned S3 on Outposts URL that can be used to perform an
 action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds that the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method,
            Params=method_parameters,
            ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.",
 client_method)
```

```
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('-'*88)
    print("Welcome to the Amazon S3 on Outposts presigned URL demo.")
    print('-'*88)

    parser = argparse.ArgumentParser()
    parser.add_argument('accessPointArn', help="The name of the S3 on Outposts
 access point ARN.")
    parser.add_argument(
        'key', help="For a GET operation, the key of the object in S3 on
 Outposts. For a "
                    "PUT operation, the name of a file to upload.")
    parser.add_argument(
        'action', choices=('get', 'put'), help="The action to perform.")
    args = parser.parse_args()

    s3_client = boto3.client('s3')
    client_action = 'get_object' if args.action == 'get' else 'put_object'
    url = generate_presigned_url(
        s3_client, client_action, {'Bucket': args.accessPointArn, 'Key':
 args.key}, 1000)

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == 'get':
        response = requests.get(url)
    elif args.action == 'put':
        print("Putting data to the URL.")
        try:
            with open(args.key, 'r') as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(f"Couldn't find {args.key}. For a PUT operation, the key must
 be the "
                  f"name of a file that exists on your computer.")

    if response is not None:
```

```
        print("Got response:")
        print(f"Status: {response.status_code}")
        print(response.text)

    print('-'*88)


if __name__ == '__main__':
    usage_demo()
```

ローカルの Amazon EMR on Outposts を使用した Amazon S3 on Outposts

Amazon EMR は、Apache Hadoop や Apache Spark などのビッグデータフレームワークを AWS で簡単に実行して、膨大な量のデータを処理および分析できるマネージドクラスタープラットフォームです。これらのフレームワークと、関連するオープンソースプロジェクトを使用することで、分析用のデータやビジネスインテリジェンスワークロードを処理できます。Amazon EMR は、大量のデータを変換したり、他の AWS データストアやデータベースとの間で出し入れしたりできるようにします。また、Amazon S3 on Outposts をサポートしています。Amazon EMR の詳細については、「Amazon EMR 管理ガイド」の「Outposts での Amazon EMR」を参照してください。

Amazon S3 on Outposts については、Amazon EMR はバージョン 7.0.0 で Apache Hadoop S3A コネクタのサポートを開始しました。以前のバージョンの Amazon EMR はローカルの S3 on Outposts をサポートしておらず、EMR ファイルシステム (EMRFS) はサポートされていません。

サポートされているアプリケーション

Amazon S3 on Outposts を使用した Amazon EMR は、次のアプリケーションをサポートしています。

- Hadoop
- Spark
- Hue
- Hive
- Sqoop
- Pig
- Hudi
- Flink

詳細については、「Amazon EMR リリースガイド」を参照してください。

Amazon S3 on Outposts バケットを作成して設定する

Amazon EMR では、Amazon S3 on Outposts で AWS SDK for Java を使用して、入力データと出力データを保存します。Amazon EMR ログファイルは、選択したリージョンの Amazon S3 ロケーションに保存され、Outpost にローカルに保存されることはありません。詳細については、「Amazon EMR 管理ガイド」の「Amazon EMR ログ」を参照してください。

Amazon S3 と DNS の要件に従うには、S3 on Outposts バケットに命名に関する制約と制限があります。詳細については、「S3 on Outposts バケットを作成する」を参照してください。

Amazon EMR バージョン 7.0.0 以降では、S3 on Outposts と S3A ファイルシステムで Amazon EMR を使用できます。

前提条件

S3 on Outposts アクセス許可 – Amazon EMR インスタンスプロファイルを作成する場合、ロールには S3 on Outposts の AWS Identity and Access Management (IAM) 名前空間が含まれている必要があります。S3 on Outposts には独自の名前空間、`s3-outposts*` があります。この名前空間を使用するポリシーの例については、「S3 on Outposts で IAM を設定する」を参照してください。

S3A コネクタ – Amazon S3 on Outposts バケットのデータにアクセスするように EMR クラスターを設定するには、Apache Hadoop S3A コネクタを使用する必要があります。コネクタを使用するには、すべての S3 URI が `s3a` スキームを使用していることを確認してください。そうでない場合は、S3 URI が S3A コネクタと連携するように EMR クラスターに使用するファイルシステムの実装を設定できます。

S3A コネクタと連携するようにファイルシステムの実装を設定するには、EMR クラスターの `fs.file_scheme.impl` および `fs.AbstractFileSystem.file_scheme.impl` 設定プロパティを使用します。*`file_scheme`* は使用している S3 URI のタイプに対応します。次の例を使用するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。例えば、`s3` スキームを使用する S3 URI ファイルシステムの実装を変更するには、次のクラスター設定プロパティを指定します。

```
[
  {
"Classification": "core-site",
    "Properties": {
    "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
    "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
```

```
    }
  }
]
```

S3A を使用するには、fs.*file_scheme*.impl 設定プロパティを org.apache.hadoop.fs.s3a.S3AFileSystem に設定し、fs.AbstractFileSystem.*file_scheme*.impl プロパティを org.apache.hadoop.fs.s3a.S3A に設定します。

例えば、パス s3a://*bucket*/... にアクセスする場合は、fs.s3a.impl プロパティを org.apache.hadoop.fs.s3a.S3AFileSystem に設定し、fs.AbstractFileSystem.s3a.impl プロパティを org.apache.hadoop.fs.s3a.S3A に設定します。

Amazon S3 on Outposts での Amazon EMR の開始方法

以下のトピックでは、Amazon S3 on Outposts で Amazon EMR の使用を開始する方法について説明します。

トピック

- 許可ポリシーを作成する
- クラスターを作成および設定する
- 設定の概要
- 考慮事項

許可ポリシーを作成する

Amazon S3 on Outposts を使用する EMR クラスターを作成する前に、クラスターの Amazon EC2 インスタンスプロファイルにアタッチする IAM ポリシーを作成する必要があります。ポリシーには、S3 on Outposts アクセスポイントの Amazon リソースネーム (ARN) にアクセスするためのアクセス許可が必要です。S3 on Outposts の IAM ポリシーの作成に関する詳細については、「S3 on Outposts で IAM を設定する」を参照してください。

次のポリシー例は、必要なアクセス許可を付与する方法を示します。ポリシーを作成したら、the section called “クラスターを作成および設定する” セクションで説明されているように、お使いの EMR クラスターの作成に使用するインスタンスプロファイルロールにポリシーをアタッチします。この例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
"Version":"2012-10-17",
  "Statement": [
        {
  "Effect": "Allow",
            "Resource": "arn:aws:s3-outposts:us-
west-2:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/access-point-name,
            "Action": [
                "s3-outposts:*"
            ]
        }
    ]

 }
```

クラスターを作成および設定する

S3 on Outposts で Spark を実行するクラスターを作成するには、コンソールで次の手順を実行します。

S3 on Outposts で Spark を実行するクラスターを作成するには

1. Amazon EMR コンソール (https://console.aws.amazon.com/elasticmapreduce/) を開きます。
2. 左のナビゲーションペインで [クラスター] を選択します。
3. [クラスターを作成] を選択します。
4. [Amazon EMR リリース] で、emr-7.0.0 以降を選択します。
5. アプリケーションバンドル で、[Spark インタラクティブ] を選択します。次に、クラスターに含める他のサポートされているアプリケーションを選択します。
6. Amazon S3 on Outposts を有効にするには、構成設定を入力します。

   構成設定の例

   次の構成設定の例を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

   ```
   [
    {
      "Classification": "core-site",
      "Properties": {
   ```

```
     "fs.s3a.bucket.DOC-EXAMPLE-BUCKET.accesspoint.arn": "arn:aws:s3-outposts:us-
west-2:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/access-point-name"
     "fs.s3a.committer.name": "magic",
     "fs.s3a.select.enabled": "false"
    }
  },
  {
    "Classification": "hadoop-env",
    "Configurations": [
      {
        "Classification": "export",
        "Properties": {
          "JAVA_HOME": "/usr/lib/jvm/java-11-amazon-corretto.x86_64"
          }
       }
     ],
     "Properties": {}
   },
   {
     "Classification": "spark-env",
     "Configurations": [
       {
         "Classification": "export",
         "Properties": {
           "JAVA_HOME": "/usr/lib/jvm/java-11-amazon-corretto.x86_64"
         }
       }
      ],
      "Properties": {}
     },
     {
      "Classification": "spark-defaults",
      "Properties": {
        "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-11-amazon-
corretto.x86_64",
        "spark.sql.sources.fastS3PartitionDiscovery.enabled": "false"
      }
     }
  ]
```

7. [ネットワーク] セクションで、AWS Outposts ラック上にある仮想プライベートクラウド (VPC) とサブネットを選択します。Amazon EMR on Outposts の詳細については、「Amazon EMR 管理ガイド」の「AWS Outposts での EMR クラスター」を参照してください。
8. [Amazon EMR の EC2 インスタンスプロファイル] セクションで、前に作成したアクセス許可ポリシーがアタッチされている IAM ロールを選択します。
9. 残りのクラスター設定を行い、[クラスターの作成] を選択します。

設定の概要

次の表は、Amazon EMR で S3 on Outposts を使用するクラスターをセットアップするときにパラメータに指定する S3A および Spark 設定と値を示しています。

S3A 設定

パラメータ	デフォルト値	S3 on Outposts に必要な値	説明
`fs.s3a.aws.credentials.provider`	指定しない場合、S3A は Outposts のバケット名を持つリージョンバケットで S3 を検索します。	S3 on Outposts バケットのアクセスポイント ARN	Amazon S3 on Outposts は、Outposts バケットにアクセスする唯一の手段として、Virtual Private Cloud (VPC) のみのアクセスポイントをサポートします。
`fs.s3a.committer.name`	`file`	`magic`	マジックコミッターは、S3 on Outposts でサポートされている唯一のコミッターです。
`fs.s3a.select.enabled`	`TRUE`	`FALSE`	S3 Select は Outposts ではサポートされていません。
`JAVA_HOME`	`/usr/lib/jvm/java-8`	`/usr/lib/jvm/java-11-amazon`	S3A 上の S3 on Outposts には、Java

パラメータ	デフォルト値	S3 on Outposts に必要な値	説明
		`-corretto .x86_64`	バージョン 11 が必要です。

Spark の設定

パラメータ	デフォルト値	S3 on Outposts に必要な値	説明
`spark.sql .sources. fastS3Par titionDis covery.enabled`	`TRUE`	`FALSE`	S3 on Outposts は高速パーティションをサポートしていません。
`spark.exe cutorEnv. JAVA_HOME`	`/usr/lib/jvm/ java-8`	`/usr/lib/jvm/ java-11-amazon -corretto .x86_64`	S3A 上の S3 on Outposts には、Java バージョン 11 が必要です。

考慮事項

Amazon EMR を S3 on Outposts バケットと統合するときは、次の点を考慮してください。

- Amazon S3 on Outposts は、Amazon EMR バージョン 7.0.0 以降でサポートされています。
- Amazon EMR で S3 on Outposts を使用するには、S3A コネクタが必要です。S3 on Outposts バケットを操作するために必要な機能を備えているのは S3A のみです。S3A コネクタのセットアップ情報については、「前提条件」を参照してください。
- Amazon S3 on Outposts は、Amazon EMR での Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) のみをサポートします。詳細については、「the section called “データ暗号化”」を参照してください。
- Amazon S3 on Outposts は、S3A FileOutputCommitter を使用した書き込みをサポートしていません。S3 on Outposts バケットに S3A FileOutputCommitter を使用して書き込むと、次のエラーが発生します。InvalidStorageClass: The storage class you specified is not valid

- Amazon S3 on Outposts は、Amazon EMR Serverless または Amazon EMR on EKS ではサポートされていません。
- Amazon EMR ログは、選択したリージョンの Amazon S3 ロケーションに保存され、S3 on Outposts バケットにローカルに保存されることはありません。

認可と認証キャッシング

S3 on Outposts は、Outposts ラック上のローカルで認証および認可データを安全にキャッシュします。キャッシュは、リクエストごとに親 AWS リージョン へのラウンドトリップを削除します。これにより、ネットワークラウンドトリップに伴う変動を排除できます。S3 on Outposts の認証キャッシュと認可キャッシュを使用すると、Outposts と AWS リージョン 間の接続のレイテンシーとは無関係の一貫したレイテンシーが得られます。

S3 on Outposts API リクエストを行うと、認証および認可データは安全にキャッシュされます。その後、キャッシュされたデータは、後続の S3 オブジェクト API リクエストを認証するために使用されます。S3 on Outposts は、リクエストが Signature Version 4A (SigV4A) を使用して署名された場合にのみ、認証および認可データをキャッシュします。キャッシュは、S3 on Outposts サービス内の Outposts にローカルに保存されます。S3 API リクエストを行うと、非同期的に更新されます。キャッシュは暗号化され、プレーンテキストの暗号化キーは Outposts に保存されません。

キャッシュは、Outpost が AWS リージョン に接続されてから最大 10 分間有効です。S3 on Outposts API リクエストを行うと、最新のポリシーが使用されるように非同期的に更新されます。Outpost が AWS リージョン から切断されている場合、キャッシュは最大 12 時間有効です。

認証キャッシュと認可キャッシュの設定

S3 on Outposts は、SigV4A アルゴリズムで署名されたリクエストの認証および認可データを自動的にキャッシュします。詳細については、「AWS Identity and Access Management ユーザーガイド」の「AWS API リクエストの署名」を参照してください。SigV4A アルゴリズムは、最新バージョンの AWS SDK で使用できます。これは、AWSCommon Runtime (CRT) ライブラリへの依存関係を通じて取得できます。

最新バージョンの AWS SDK を使用し、最新バージョンの CRT をインストールする必要があります。例えば、`pip install awscrt` を実行して Boto3 で CRT の最新バージョンを取得できます。

S3 on Outposts は、SigV4 アルゴリズムで署名されたリクエストの認証および認可データをキャッシュしません。

SigV4A 署名の検証

AWS CloudTrail を使用してリクエストが SigV4A で署名されたことを確認できます。S3 on Outposts の CloudTrail の設定についての詳細は、「AWS CloudTrail ログで S3 on Outposts をモニタリングする」を参照してください。

CloudTrail を設定したら、CloudTrail ログの `SignatureVersion` フィールドでリクエストがどのように署名されたかを確認できます。SigV4A で署名されたリクエストでは、`SignatureVersion` が `AWS4-ECDSA-P256-SHA256` に設定されます。SigV4 で署名されたリクエストでは、`SignatureVersion` が `AWS4-HMAC-SHA256` に設定されます。

S3 on Outposts のセキュリティ

AWS では、クラウドセキュリティを最優先事項としています。AWS のユーザーは、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを利用できます。

セキュリティは、AWS とユーザーの間の責任共有です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ — AWS は、AWS クラウド で AWS のサービス を実行するインフラストラクチャを保護する責任を負います。また AWS は、お客様が使用するサービスを安全に提供します。AWS コンプライアンスプログラム の一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon S3 on Outposts に適用するコンプライアンスプログラムの詳細については、「コンプライアンスプログラムによる対象範囲の AWS サービス」を参照してください。
- クラウド内のセキュリティ — お客様の責任は、使用する AWS のサービス に応じて異なります。また、お客様は、お客様のデータの機密性、企業の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、S3 on Outposts を使用する際に責任共有モデルを適用する方法を理解するのに役立ちます。以下のトピックでは、セキュリティとコンプライアンスの目標を満たすように S3 on Outposts を設定する方法について説明します。また、S3 on Outposts リソースのモニタリングと安全確保に役立つその他の AWS のサービス の使用方法も説明します。

トピック

- S3 on Outposts のデータ暗号化
- S3 on Outposts の AWS PrivateLink

- AWS Signature Version 4 (SigV4) での認証固有ポリシーキー
- Amazon S3 on Outposts 用の AWS マネージドポリシー
- Amazon S3 on Outposts でのサービスにリンクされたロールの使用

S3 on Outposts のデータ暗号化

デフォルトでは、Amazon S3 on Outposts に保存されるすべてのデータは、Amazon S3 マネージド暗号化キーによるサーバー側の暗号化 (SSE-S3) を使用して暗号化されます。詳細については、「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)」を参照してください。

オプションで、ユーザーが用意した暗号化キーによるサーバー側の暗号化 (SSE-C) を使用できます。SSE-C を使用するには、オブジェクト API リクエストの一部として暗号化キーを指定します。サーバー側の暗号化では、オブジェクトのメタデータではなく、オブジェクトデータのみが暗号化されます。詳細については、「お客様が指定したキーによるサーバー側の暗号化 (SSE-C) の使用」を参照してください。

> **Note**
>
> S3 on Outposts は、AWS Key Management Service (AWS KMS) キー (SSE-KMS) を使用したサーバー側暗号化をサポートしていません。

S3 on Outposts の AWS PrivateLink

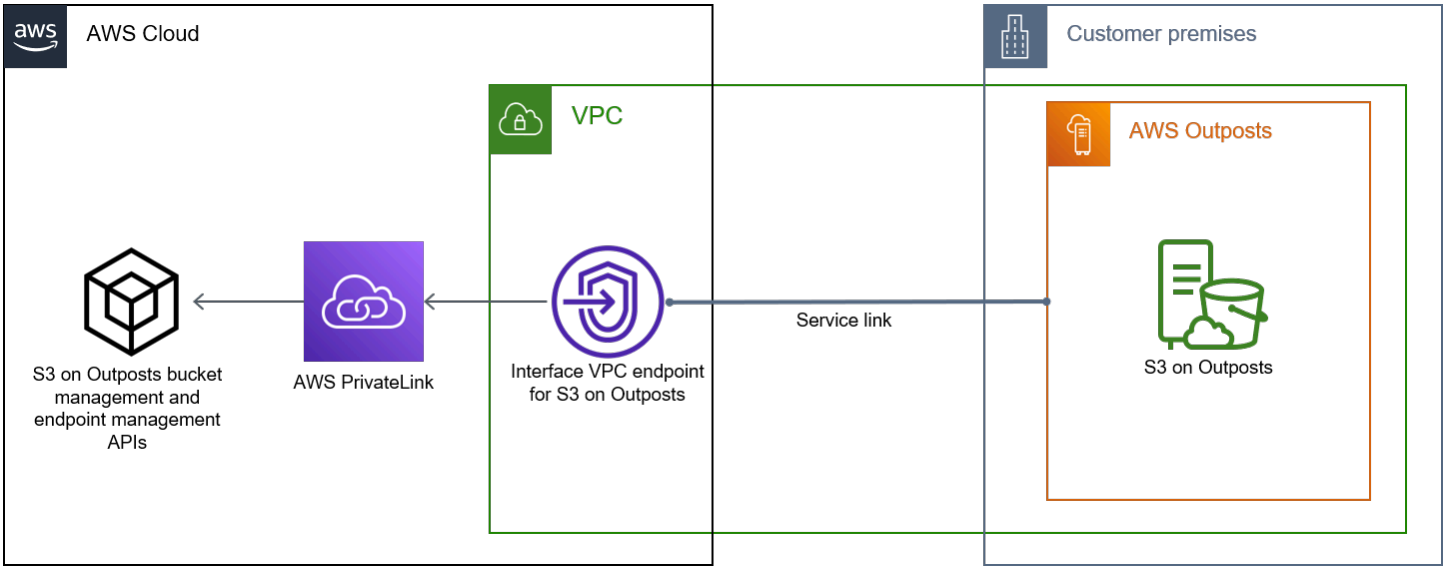
S3 on Outposts は AWS PrivateLink をサポートしているため、仮想プライベートネットワーク内のプライベートエンドポイント経由で S3 on Outposts ストレージに直接管理アクセスを提供します。これにより、仮想プライベートクラウド (VPC) のプライベート IP アドレスを使用して、内部ネットワークアーキテクチャを簡素化し、Outposts オブジェクトストレージに対して管理オペレーションを実行できます。AWS PrivateLink を使用することで、パブリック IP アドレスやプロキシサーバーを使用する必要がなくなります。

Amazon S3 on Outposts の AWS PrivateLink では、インターフェイス VPC エンドポイントを仮想プライベートクラウド (VPC) でプロビジョニングし、S3 on Outposts のバケット管理およびエンドポイント管理 API にアクセスできます。インターフェイス VPC エンドポイントには、VPC またはオンプレミスにデプロイしたアプリケーションから、仮想プライベートネットワーク (VPN) や AWS Direct Connect 経由で直接アクセスできます。バケット管理 API とエンドポイント管理 API には、AWS PrivateLink を介してアクセスできます。AWS PrivateLink は、データ転送 API オペレー

ション (GET、PUT、同様の API など) をサポートしていません。これらのオペレーションは、S3 on Outposts エンドポイントとアクセスポイントの設定を通じて既にプライベートに転送されています。詳細については、「S3 on Outposts のネットワーキング」を参照してください。

インターフェイスエンドポイントは、VPC 内のサブネットからプライベート IP アドレスが割り当てられた 1 つ以上の Elastic Network Interface (ENI) で表されます。S3 on Outposts のインターフェイスエンドポイントに対するリクエストは、AWS ネットワーク上の S3 on Outposts バケット管理およびエンドポイント管理 API に自動的にルーティングされます。AWS Direct Connect または AWS Virtual Private Network (AWS VPN) を介して、オンプレミスのアプリケーションから VPC 内のインターフェイスエンドポイントにアクセスすることもできます。VPC をオンプレミスネットワークに接続する方法の詳細については、AWS Direct Connect ユーザーガイドおよび AWS Site-to-Site VPN ユーザーガイドを参照してください。

次の図に示すように、インターフェイスエンドポイントは、AWS ネットワークおよび AWS PrivateLink を介してリクエストを S3 on Outposts のバケット管理およびエンドポイント管理 API にルーティングします。



インターフェイスエンドポイントの一般的な情報については、AWS PrivateLink ガイドのインターフェイス VPC エンドポイント (AWS PrivateLink) を参照してください。

トピック

- 制約と制限
- S3 on Outposts インターフェイスエンドポイントへのアクセス
- オンプレミスの DNS 設定の更新
- S3 on Outposts 用の VPC エンドポイントの作成

- S3 on Outposts のバケットポリシーと VPC エンドポイントポリシーの作成

制約と制限

AWS PrivateLink を介して S3 on Outposts のバケット管理およびエンドポイント管理 API にアクセスする場合は、VPC の制限が適用されます。詳細については、AWS PrivateLink ガイドの インターフェイスエンドポイントのプロパティと制限 と AWS PrivateLink クォータ を参照してください。

また、AWS PrivateLink では、以下の機能はサポートされていません。

- 連邦情報処理規格 (FIPS) エンドポイント
- S3 on Outposts データ転送 API、GET、PUT、および類似のオブジェクト API オペレーションなど。
- プライベート DNS

S3 on Outposts インターフェイスエンドポイントへのアクセス

AWS PrivateLink を使用して S3 on Outposts バケット管理およびエンドポイント管理 API にアクセスするには、エンドポイント固有の DNS 名を使用するようにアプリケーションを更新する必要があります。インターフェイスエンドポイントを作成すると、AWS PrivateLink は、Amazon S3 はエンドポイント固有の 2 つのタイプの S3 on Outposts 名 (Regional および zonal) を生成します。

- Regional DNS 名 – 一意の VPC エンドポイント ID、サービス識別子、AWS リージョン、および `vpce.amazonaws.com` 、*`vpce-1a2b3c4d-5e6f`*`.s3-outposts.us-east-1.vpce.amazonaws.com` などが含まれます。
- Zonal DNS 名— 一意の VPC エンドポイント ID、アベイラビリティーゾーン、サービス識別子、AWS リージョン、および `vpce.amazonaws.com` たとえば *`vpce-1a2b3c4d-5e6f`*`-us-east-1a.s3-outposts.us-east-1.vpce.amazonaws.com` が含まれます。このオプションは、アーキテクチャがアベイラビリティーゾーンを分離する場合に使用できます。例えば、障害を隔離し、リージョン間のデータ転送コストを削減するために Zonal DNS 名を使用できます。

> **Important**
>
> S3 on Outposts のインターフェイスエンドポイントは、パブリック DNS ドメインから解決されます。S3 on Outposts はプライベート DNS をサポートしていません。すべてのバケット管理およびエンドポイント管理 API には `--endpoint-url` パラメータを使用します。

AWS CLI の例

`--region` および `--endpoint-url` パラメータを使用して S3 on Outposts インターフェイスエンドポイントを介してバケット管理およびエンドポイント管理 API にアクセスします。

Example : エンドポイント URL を使用して S3 コントロール API のバケットをリスト化します。

次の例では、リージョン *us-east-1*、VPC エンドポイント URL *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*、およびアカウント ID *111122223333* を適切な情報に置き換えます。

```
aws s3control list-regional-buckets --region us-east-1 --endpoint-url
 https://vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com --account-
id 111122223333
```

AWS SDK の例

S3 on Outpostsインターフェイスエンドポイントの S3 コントロール API にアクセスするように、SDK を最新バージョンに更新して、エンドポイント URL を使用するようにクライアントを設定します。詳細については、「AWSSDKの例AWS PrivateLink」を参照してください。

SDK for Python (Boto3)

Example : エンドポイント URL を使用して S3 コントロール API にアクセスする

次の例では、リージョン *us-east-1*と VPC エンドポイント URLURL *vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com* を適切な情報に置き換えます。

```
control_client = session.client(
service_name='s3control',
region_name='us-east-1',
endpoint_url='https://vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com'
)
```

詳細については、Boto3 デベロッパーガイドの「Amazon S3 の AWS PrivateLink」を参照してください。

SDK for Java 2.x

Example : エンドポイント URL を使用して S3 コントロール API にアクセスする

次の例では、VPC エンドポイント URL *vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com* とリージョン *Region.US_EAST_1* を適切な情報に置き換えます。

```
// control client
Region region = Region.US_EAST_1;
s3ControlClient = S3ControlClient.builder().region(region)

 .endpointOverride(URI.create("https://vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com"))
                                  .build()
```

詳細については、AWS SDK for Java API リファレンスの「S3ControlClient」を参照してください。

オンプレミスの DNS 設定の更新

エンドポイント固有の DNS 名を使用して S3 on Outposts バケット管理およびエンドポイント管理 API にアクセスする場合、オンプレミス DNS リゾルバーを更新する必要はありません。パブリック S3 on Outposts DNS ドメインからのインターフェイスエンドポイントのプライベート IP アドレスを使用して、エンドポイント固有の DNS 名を解決できます。

S3 on Outposts 用の VPC エンドポイントの作成

S3 on Outposts の VPC インターフェイスエンドポイントを作成するには、「AWS PrivateLink ガイド」の「VPC エンドポイントの作成」を参照してください。

S3 on Outposts のバケットポリシーと VPC エンドポイントポリシーの作成

VPC エンドポイントに S3 on Outposts へのアクセスをコントロールするエンドポイントポリシーをアタッチできます。また、S3 on Outposts バケットポリシーの `aws:sourceVpce` 条件を使用して、特定の VPC エンドポイントからの特定のバケットへのアクセスを制限することもできます。VPC エンドポイントポリシーを使用すると、S3 on Outposts バケット管理 API およびエンドポイント管理 API へのアクセスを制御できます。バケットポリシーを使用すると、S3 on Outposts バケット管理 API へのアクセスを制御できます。ただし、`aws:sourceVpce` を使用して S3 on Outposts のオブジェクトアクションへのアクセスを管理することはできません。

S3 on Outposts のアクセスポリシーでは、以下の情報が指定されています。

- アクションを許可または拒否する AWS Identity and Access Management (IAM) プリンシパル。
- 許可または拒否されるS3 コントロールアクション。
- アクションが許可または拒否される S3 on Outposts リソース。

次の例は、バケットまたはエンドポイントへのアクセスを制限するポリシーを示しています。VPC 接続の詳細については、 ホワイトペーパー Amazon Virtual Private Cloud 接続オプションAWS のネットワークから VPC への接続オプション を参照してください。

> **Important**
>
> - このセクションで説明している VPC エンドポイントに例のポリシーを適用すると、バケットへのアクセスが意図せずにブロックされる場合があります。VPC エンドポイントからの接続に対するバケットのアクセスを制限するバケットのアクセス許可により、バケットへのすべての接続がブロックされる場合があります。この問題を修正する方法については、バケットポリシーの VPC または VPC エンドポイント ID が間違っています。「ポリシーを修正してバケットにアクセスできるようにするにはどうすれば良いですか?」(AWS Support ナレッジセンター) を参照してください。
> - 次のバケットポリシーの例を使用する前に、VPC エンドポイントの ID をユースケースに応じた値に置き換えてください。そうしないと、バケットにアクセスできません。
> - ポリシーで、特定の VPC エンドポイントからの S3 on Outposts バケットへのアクセスだけが許可されている場合、コンソールリクエストは指定の VPC エンドポイントから送信されないため、そのバケットのコンソールアクセスは無効になります。

トピック

- 例: VPC エンドポイントから特定のバケットへのアクセスの制限
- 例:S3 on Outposts バケットポリシーの特定の VPC エンドポイントからのアクセスを拒否する

例: VPC エンドポイントから特定のバケットへのアクセスの制限

特定の S3 on Outposts バケットへのアクセスのみを制限するエンドポイントポリシーを作成できます。次のポリシーは、GetBucketPolicy アクションのアクセスを *example-outpost-bucket* のみに制限します。このポリシーを使用するには、例の値を独自の値に置き換えます。

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909151",
  "Statement": [
    { "Sid": "Access-to-specific-bucket-only",
      "Principal": {"AWS":"111122223333"},
      "Action": "s3-outposts:GetBucketPolicy",
      "Effect": "Allow",
      "Resource": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outpost-
bucket"
    }
  ]
}
```

例:S3 on Outposts バケットポリシーの特定の VPC エンドポイントからのアクセスを拒否する

次の S3 on Outposts バケットポリシーは、*vpce-1a2b3c4d*VPC エンドポイントを使用する *example-outpost-bucket* バケットの GetBucketPolicy へのアクセスを拒否します。

aws:sourceVpce 条件はエンドポイントを指定し、VPC エンドポイントリソースの Amazon リソースネーム (ARN) を必要とせず、エンドポイント ID のみを指定します。このポリシーを使用するには、例の値を独自の値に置き換えます。

```
{
    "Version": "2012-10-17",
    "Id": "Policy1415115909152",
    "Statement": [
        {
            "Sid": "Deny-access-to-specific-VPCE",
            "Principal": {"AWS":"111122223333"},
            "Action": "s3-outposts:GetBucketPolicy",
            "Effect": "Deny",
            "Resource": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outpost-
bucket",
            "Condition": {
                "StringEquals": {"aws:sourceVpce": "vpce-1a2b3c4d"}
            }
        }
    ]
}
```

AWS Signature Version 4 (SigV4) での認証固有ポリシーキー

次の表に、Amazon S3 on Outposts で使用する AWS Signature Version 4 (SigV4) 認証に関連する条件キーを示します。これらの条件をバケットポリシーに追加して、Signature Version 4 を使用してリクエストが認証された際の特定の動作を適用できます。エンドポイントポリシーの例については、「Signature Version 4 関連の条件キーを使用するバケットポリシーの例」を参照してください。Signature Version 4 によるリクエストの認証の詳細については、「Amazon Simple Storage Service API リファレンス」の「Authenticating requests (AWS Signature Version 4)」(リクエストの認証 (AWS Signature Version 4)) を参照してください。

`s3-outposts:*` アクションまたは S3 on Outposts アクションに適用可能なキー

適用できるキー	説明
`s3-outposts:authType`	S3 on Outposts では、さまざまな認証方式をサポートしています。受信するリクエストが特定の認証方法を使用するように制限するには、このオプションの条件キーを使用します。例えば、この条件キーを使用すると、リクエスト認証のために HTTP `Authorization` ヘッダーのみの使用を指定できます。 有効な値: `REST-HEADER` `REST-QUERY-STRING`
`s3-outposts:signatureAge`	認証されたリクエストの中で署名が有効である時間長 (ミリ秒単位)。 この条件は、署名付き URL でのみ機能します。 Signature Version 4 では、署名キーは最大 7 日間有効です。したがって、署名の有効期間も最大 7 日間となります。詳細については、「Amazon Simple Storage Service API リファレンス」の「Introduction to Signing Requests」(リクエスト署名の導入) を参照してください。この条件を使用すると、署名の有効期間をさらに制限できます。 値の例: `600000`

適用できるキー	説明
s3-outposts:x-amz-content-sha256	この条件キーを使用すると、バケット内にある署名されていないコンテンツを許可しません。 Authorization ヘッダを使用するリクエストに対し Signature Version 4 を使用する場合、署名計算に x-amz-content-sha256 ヘッダーを追加した後、その値をハッシュペイロードに設定します。 この条件キーをバケットポリシーで使用すると、ペイロードに署名がないアップロードを、すべて拒否することができます。例: • リクエスト認証において、Authorization ヘッダーを使用するアップロードを拒否し、ペイロードには署名しません。詳細については、「Amazon Simple Storage Service API リファレンス」の「Transferring payload in a single chunk」(ペイロードを 1 つのチャンクで転送する) を参照してください。 • 署名付き URL を使用するアップロードを拒否します。署名済み URL は常に UNSIGNED_PAYLOAD を持ちます。詳細については、「Amazon Simple Storage Service API リファレンス」の「Authenticating Requests」(認証リクエスト)「Authentication methods」(認証メソッド) を参照してください。 有効な値: UNSIGNED-PAYLOAD

Signature Version 4 関連の条件キーを使用するバケットポリシーの例

次の例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

Example : **s3-outposts:signatureAge**

次のバケットポリシーは、署名が作成されてから 10分以上経過している場合、example-outpost-bucket 内のオブジェクトに対する S3 on Outposts の署名付き URL リクエストを拒否します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "Deny a presigned URL request if the signature is more than 10
 minutes old",
            "Effect": "Deny",
            "Principal": {"AWS":"444455556666"},
            "Action": "s3-outposts:*",
            "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
            "Condition": {
                "NumericGreaterThan": {"s3-outposts:signatureAge": 600000},
                "StringEquals": {"s3-outposts:authType": "REST-QUERY-STRING"}
            }
        }
    ]
}
```

Example : **s3-outposts:authType**

次のバケットポリシーでは、リクエスト認証に Authorizationヘッダーを使用しているリクエストのみを許可します。署名付き URL リクエストは、クエリパラメーターを使用してリクエストと認証情報を指定しているため、すべての署名付き URL リクエストは拒否されます。詳細については、「Amazon Simple Storage Service API リファレンス」の「Authentication methods」(認証メソッド)を参照してください。

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
               "Sid": "Allow only requests that use the Authorization header for
 request authentication. Deny presigned URL requests.",
               "Effect": "Deny",
               "Principal": {"AWS":"111122223333"},
               "Action": "s3-outposts:*",
               "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
               "Condition": {
                     "StringNotEquals": {
                           "s3-outposts:authType": "REST-HEADER"
                     }
               }
         }
   ]
```

```
}
```

Example : **s3-outposts:x-amz-content-sha256**

次のバケットポリシーは、署名付き URL を使用するアップロードなど、未署名のペイロードを含むすべてのアップロードを拒否します。詳細については、「Amazon Simple Storage Service API リファレンス」の「Authenticating Requests」(認証リクエスト)「Authentication methods」(認証メソッド) を参照してください。

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
               "Sid": "Deny uploads with unsigned payloads.",
               "Effect": "Deny",
               "Principal": {"AWS":"111122223333"},
               "Action": "s3-outposts:*",
               "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
               "Condition": {
                     "StringEquals": {
                           "s3-outposts:x-amz-content-sha256": "UNSIGNED-PAYLOAD"
                     }
               }
         }
   ]
}
```

Amazon S3 on Outposts 用の AWS マネージドポリシー

AWS マネージドポリシーは、AWS が作成および管理するスタンドアロンポリシーです。AWS マネージドポリシーは、多くの一般的なユースケースでアクセス許可を提供できるように設計されているため、ユーザー、グループ、ロールへのアクセス許可の割り当てを開始できます。

AWS マネージドポリシーは、ご利用の特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることにご注意ください。AWS のすべてのお客様が使用できるようになるのを避けるためです。ユースケース別にカスタマー管理ポリシーを定義することで、アクセス許可を絞り込むことをお勧めします。

AWS 管理ポリシーで定義したアクセス権限は変更できません。AWS が AWS マネージドポリシーに定義されているアクセス許可を更新すると、更新はポリシーがアタッチされているすべてのプリンシパルアイデンティティ (ユーザー、グループ、ロール) に影響します。新しい AWS のサービスを起動するか、既存のサービスで新しい API オペレーションが使用可能になると、AWS が AWS マネージドポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「AWS 管理ポリシー」を参照してください。

AWS マネージドポリシー: AWSS3OnOutpostsServiceRolePolicy

サービスにリンクされたロール `AWSServiceRoleForS3OnOutposts` の一部として、ネットワークリソースを管理するのに役立ちます。

このポリシーの許可を確認するには、「AWSS3OnOutpostsServiceRolePolicy」を参照してください。

AWS マネージドポリシーに対する Amazon S3 on Outposts の更新

S3 on Outposts の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。

変更	説明	日付
S3 on Outposts で `AWSS3OnOutpostsServiceRolePolicy` を追加	S3 on Outposts では、サービスにリンクされたロール `AWSServiceRoleForS3OnOutposts` の一部として `AWSS3OnOutpostsServiceRolePolicy` が追加されました。これは、ネットワークリソースを管理するのに役立ちます。	2023 年 10 月 3 日
S3 on Outposts で変更の追跡を開始	S3 on Outposts で AWS マネージドポリシーの変更の追跡を開始しました。	2023 年 10 月 3 日

Amazon S3 on Outposts でのサービスにリンクされたロールの使用

Amazon S3 on Outposts は AWS Identity and Access Management (IAM) サービスにリンクされたロールを使用します。サービスにリンクされたロールは、S3 on Outposts に直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、S3 on Outposts によって事前定義されており、ユーザーの代わりにサービスから他の AWS のサービスを呼び出す必要のあるアクセス許可がすべて含まれています。

サービスにリンクされたロールを使用することで、必要なアクセス許可を手動で追加する必要がなくなるため、S3 on Outposts の設定が簡単になります。S3 on Outposts は、サービスにリンクされたロールのアクセス許可を定義します。特に定義されている場合を除き、S3 on Outposts のみがそのロールを引き受けることができます。定義される許可には、信頼ポリシーと許可ポリシーが含まれており、その許可ポリシーを他のIAM エンティティにアタッチすることはできません。

サービスにリンクされたロールは、まずその関連リソースを削除しなければ削除できません。これにより、リソースへのアクセス許可を誤って削除できないようになるため、S3 on Outposts リソースが保護されます。

サービスにリンクされたロールをサポートする他のサービスについては、「IAM と連動する AWS のサービス」を参照し、[Service-linked roles] (サービスにリンクされたロール) の列内で [Yes] (はい) と表記されたサービスを確認してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

S3 on Outposts のサービスにリンクされたロールのアクセス許可

S3 on Outposts は、AWSServiceRoleForS3OnOutposts という名前のロールを使用して、ユーザーに代わってネットワークリソースを管理します。

`AWSServiceRoleForS3OnOutposts` サービスにリンクされたロールは、ロールの引き受けについて以下のサービスを信頼します。

- `s3-outposts.amazonaws.com`

`AWSS3OnOutpostsServiceRolePolicy` というロールアクセス許可ポリシーは、S3 on Outposts に、指定されたリソースで次のアクションを完了することを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DescribeCoipPools",
                "ec2:GetCoipPoolUsage",
                "ec2:DescribeAddresses",
                "ec2:DescribeLocalGatewayRouteTableVpcAssociations"
            ],
            "Resource": "*",
            "Sid": "DescribeVpcResources"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ],
            "Sid": "CreateNetworkInterface"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "CreateTagsForCreateNetworkInterface"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress"
            ],
```

```
            "Resource": [
                "arn:aws:ec2:*:*:ipv4pool-ec2/*"
            ],
            "Sid": "AllocateIpAddress"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:elastic-ip/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "CreateTagsForAllocateIpAddress"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DisassociateAddress",
                "ec2:ReleaseAddress",
                "ec2:AssociateAddress"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "ReleaseVpcResources"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": [
                        "CreateNetworkInterface",
                        "AllocateAddress"
                    ],
                    "aws:RequestTag/CreatedBy": [
                        "S3 On Outposts"
                    ]
                }
            },
            "Sid": "CreateTags"
        }
    ]
}
```

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ロールなど) に許可するには、アクセス許可を設定する必要があります。詳細については、IAM ユーザーガイドの「サービスリンクロールのアクセス許可」を参照してください。

S3 on Outposts でのサービスにリンクされたロールの作成

サービスにリンクされたロールを手動で作成する必要はありません。AWS Management Console、AWS CLI、または AWS API で S3 on Outposts エンドポイントを作成すると、S3 on Outposts によってサービスにリンクされたロールが作成されます。

このサービスにリンクされたロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。S3 on Outposts エンドポイントを作成すると、S3 on Outposts によってサービスにリンクされたロールが再び作成されます。

IAM コンソールを使用して、S3 on Outposts ユースケースでサービスにリンクされたロールを作成することもできます。AWS CLI または AWS API で、`s3-outposts.amazonaws.com` サービス名を使用してサービスリンクロールを作成します。詳細については、IAM ユーザーガイドの「サービスリンクロールの作成」を参照してください。このサービスリンクロールを削除する場合、この同じプロセスを使用して、もう一度ロールを作成できます。

S3 on Outposts でのサービスにリンクされたロールの編集

S3 on Outposts では、サービスにリンクされたロール `AWSServiceRoleForS3OnOutposts` を編集することはできません。さまざまなエンティティが参照する可能性があるため、これにはロール

名が含まれます。ただし、IAM を使用したロールの説明の編集はできます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

S3 on Outposts でのサービスにリンクされたロールの削除

サービスにリンクされたロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、積極的にモニタリングまたは保守されていない未使用のエンティティを排除できます。ただし、手動で削除する前に、サービスにリンクされたロールのリソースをクリーンアップする必要があります。

> **Note**
>
> リソースを削除する際に、S3 on Outposts サービスでそのロールが使用されている場合、削除は失敗することがあります。失敗した場合は、数分待ってから操作を再試行してください。

AWSServiceRoleForS3OnOutposts が使用する S3 on Outposts リソースを削除するには

1. すべての AWS リージョンの AWS アカウントで、Outposts エンドポイントの S3 を削除します。
2. IAM を使用して、サービスにリンクされたロールを削除します。

   IAM コンソール、AWS CLI、または AWS API を使用して、`AWSServiceRoleForS3OnOutposts` サービスリンクロールを削除します。詳細については、IAM ユーザーガイドの「サービスにリンクされたロールの削除」を参照してください。

S3 on Outposts でのサービスにリンクされたロールをサポートするリージョン

S3 on Outposts は、そのサービスを利用できるすべての AWS リージョンで、サービスにリンクされたロールの使用をサポートします。詳細については、「S3 on Outposts のリージョンとエンドポイント」を参照してください。

S3 on Outposts ストレージの管理

Amazon S3 on Outposts を使用すると、AWS Outposts で S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレ

ミスのオブジェクトを簡単に保存および取得できます。S3 on Outposts は、新しいストレージクラス、S3 Outposts (OUTPOSTS) を提供し、これは Amazon S3 API を使用し、AWS Outposts 上の複数のデバイスやサーバー間でデータを永続的かつ冗長的に保存するように設計されています。仮想プライベートクラウド (VPC) を介したアクセスポイントとエンドポイント接続を使用して、Outposts バケットと通信します。Outposts バケットでは、Amazon S3 と同じ API と機能 (アクセスポリシー、暗号化、タグ付けなど) を使用できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。詳細については、「Amazon S3 on Outposts とは」を参照してください。

Amazon S3 on Outposts ストレージ容量の管理および共有の詳細については、以下のトピックを参照してください。

トピック

- S3 on Outposts バケットの S3 バージョニングの管理
- Amazon S3 on Outposts バケットのライフサイクル設定を作成および管理する
- S3 on Outposts のオブジェクトのレプリケート
- AWS RAM を使用して S3 on Outposts を共有する
- S3 on Outposts を使用するその他の AWS のサービス

S3 on Outposts バケットの S3 バージョニングの管理

有効にすると、S3 バージョニングは、同じバケット内にあるオブジェクトの複数の異なるコピーを保存します。S3 バージョニングを使用して、Outposts バケットに保存されたあらゆるオブジェクトのあらゆるバージョンを保存、取得、復元することができます。S3 バージョニングによって、意図しないユーザーアクションやアプリケーション障害から復旧できます。

Amazon S3 on Outposts バケットには、次の 3 つのバージョニング状態があります。

- バージョニングが無効 — バケットの S3 バージョニングを有効化または停止したことがない場合は、バージョニングが無効であり、S3 バージョニングの状態は返されません。S3 バージョニングの詳細については、S3 バケットでのバージョニングの使用 を参照してください。
- 有効 — バケット内のオブジェクトの S3 バージョニングを有効にします。バケットに追加されたすべてのオブジェクトは、一意のバージョン ID を受け取ります。バージョニングを有効にした時点でバケットにすでに存在していたオブジェクトのバージョン ID は `null` です。これらの (またはその他の) オブジェクトを他のオペレーション (PutObject) で変更すると、新しいオブジェクトは一意のバージョン ID を取得します。

- 停止 — バケット内のオブジェクトの S3 バージョニングを停止します。バージョニングが停止された後に、バケットに追加されたすべてのオブジェクトは、バージョン ID null を受け取ります。詳細については、「バージョニングが停止されたバケットへのオブジェクトの追加」を参照してください。

S3 on Outposts バケットの S3 バージョニングを有効にすると、バージョニング無効の状態に戻すことはできません。ただし、バージョニングを停止することはできます。S3 バージョニングの詳細については、S3 バケットでのバージョニングの使用 を参照してください。

バケット内の各オブジェクトには、最新のバージョンと最新でないバージョンが 0 個以上存在します。ストレージコストを削減するには、バケットの S3 ライフサイクルルールを設定して、指定した期間の後に最新でないバージョンを期限切れにすることができます。詳細については、「Amazon S3 on Outposts バケットのライフサイクル設定を作成および管理する」を参照してください。

次の例は、AWS Management Console と AWS Command Line Interface (AWS CLI) を使用して、既存の S3 on Outposts バケットのバージョニングを有効化または停止する方法を示しています。S3 バージョニングを有効にしてバケットを作成するには、「S3 on Outposts バケットを作成する」を参照してください。

> **Note**
>
> バケットを作成する AWS アカウント がそのバケットを所有し、アクションをコミットできる唯一のアカウントです。バケットには、Outposts、タグ、デフォルトの暗号化、アクセスポイント設定などの設定プロパティがあります。アクセスポイント設定には、仮想プライベートクラウド (VPC)、バケット内のオブジェクトにアクセスするためのアクセスポイントポリシー、およびその他のメタデータが含まれます。詳細については、「S3 on Outposts の仕様」を参照してください。

S3 コンソールの使用

バケットの S3 バージョニング設定を編集するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. S3 バージョニングを有効にする Outposts バケットを選択します。
4. プロパティ タブを選択します。

5. [バケットのバージョニング] で [編集] を選択します。
6. 以下のいずれかのオプションを選択して、バケットの S3 バージョニング設定を編集します。
   - S3 バージョニングを一時停止し、新しいオブジェクトバージョンの作成を停止するには、[Suspend] (停止) を選択します。
   - S3 バージョニングを有効にして、各オブジェクトの複数の異なるコピーを保存するには、[Enable] (有効化) を選択します。
7. [Save changes] (変更を保存) をクリックします。

AWS CLI の使用

AWS CLI を使用してバケットの S3 バージョニングを有効化または停止するには、次の例に示すように `put-bucket-versioning` コマンドを使用します。これらの例を実行するには、それぞれの *`user input placeholder`* をユーザー自身の情報に置き換えます。

詳細については、AWS CLI リファレンスの「put-bucket-versioning」を参照してください。

Example : S3 バージョニングを有効にするには

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --versioning-configuration Status=Enabled
```

Example : S3 バージョニングを停止するには

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --versioning-configuration Status=Suspended
```

Amazon S3 on Outposts バケットのライフサイクル設定を作成および管理する

S3 ライフサイクルを使用して、Amazon S3 on Outposts のストレージ容量を最適化することができます。ライフサイクルルールを作成して、オブジェクトが古くなったり、新しいバージョンに置き換えられたりしたときに期限切れにすることができます。ライフサイクルルールを作成、有効化、無効化、または削除できます。

S3 ライフサイクルの詳細については、「ストレージのライフサイクルの管理」を参照してください。

> **Note**
>
> バケットを作成する AWS アカウント がそのバケットを所有し、ライフサイクルルールを作成、有効化、無効化、または削除できる唯一のアカウントです。

S3 on Outposts バケットのライフサイクル設定を作成および管理するには、以下のトピックを参照してください。

トピック

- AWS Management Console を使用したライフサイクルルールの作成と管理
- AWS CLI および SDK for Java を使用したライフサイクル設定の作成と管理

AWS Management Console を使用したライフサイクルルールの作成と管理

S3 ライフサイクルを使用して、Amazon S3 on Outposts のストレージ容量を最適化することができます。ライフサイクルルールを作成して、オブジェクトが古くなったり、新しいバージョンに置き換えられたりしたときに期限切れにすることができます。ライフサイクルルールを作成、有効化、無効化、または削除できます。

S3 ライフサイクルの詳細については、「ストレージのライフサイクルの管理」を参照してください。

> **Note**
>
> バケットを作成する AWS アカウント がそのバケットを所有し、ライフサイクルルールを作成、有効化、無効化、または削除できる唯一のアカウントです。

AWS Management Console を使用して S3 on Outposts のライフサイクルルールを作成および管理するには、以下のトピックを参照してください。

トピック

- ライフサイクルルールの作成
- ライフサイクルルールの有効化
- ライフサイクルルールの編集
- ライフサイクルルールの削除

ライフサイクルルールの作成

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. ライフサイクルルールを作成する Outposts バケットを選択します。
4. [Management] (管理) タブを選択して、[Create Lifecycle rule] (ライフサイクルルールを作成する) を選択します。
5. [Lifecycle rule name] (ライフサイクルルール名) の値を入力します。
6. [Rule scope] (ルールスコープ) で、次のいずれかのオプションを選択します。

   - 特定のフィルターの適用範囲を制限するには、[Limit the scope of this rule using one or more filters] (1 つまたは複数のフィルターを使用してこのルールの適用範囲を制限) を選択します。次に、プレフィックスフィルターまたはオブジェクトサイズを追加します。
   - このルールをバケット内のすべてのオブジェクトに適用するには、[Apply to all objects in the bucket] (バケット内のすべてのオブジェクトに適用) を選択します。
7. [Lifecycle rule actions] (ライフサイクルルールのアクション) で、次のいずれかのオプションを選択します。

   - [Expire current versions of objects] (オブジェクトの現行バージョンを期限切れにする) — バージョニングが有効なバケットの場合、S3 on Outposts は削除マーカーを追加し、オブジェクトを最新でないバージョンとして保持します。S3 バージョニングを使用しないバケットの場合、S3 on Outposts はオブジェクトを完全に削除します。
   - [Permanently delete noncurrent versions of objects] (オブジェクトの最新ではないバージョンを完全に削除する) — S3 on Outposts は、オブジェクトの最新ではないバージョンを完全に削除します。
   - [Delete expired object delete markers or incomplete multipart uploads] (期限切れのオブジェクト削除マーカーまたは未完了のマルチパートアップロードを削除する) — S3 on Outposts は、期限切れのオブジェクト削除マーカーまたは未完了のマルチパートアップロードを完全に削除します。

     オブジェクトタグを使用してライフサイクルルールの範囲を制限する場合、[Delete expired object delete markers] (期限切れのオブジェクト削除マーカーを削除する) を選択することはできません。また、[Expire current object versions] (オブジェクトの現行バージョンを期限切れにする) を選択した場合は、[Delete expired object delete markers] (期限切れのオブジェクト削除マーカーを削除する) を選択することもできません。

> **Note**
>
> サイズベースのフィルターは、削除マーカーや不完全なマルチパートアップロードには使用できません。

8. [Expire current versions of objects] (オブジェクトの現行バージョンを期限切れにする) または [Permanently delete noncurrent versions of objects] (オブジェクトの最新ではないバージョンを完全に削除する) を選択した場合は、特定の日付またはオブジェクトの経過時間に基づいてルールトリガーを設定します。
9. [Delete expired object delete markers] (期限切れのオブジェクト削除マーカーを削除する) を選択した場合、期限切れのオブジェクト削除マーカーの削除を確認するには、[Delete expired object delete markers] (期限切れのオブジェクト削除マーカーを削除する) を選択します。
10. [Timeline Summary] (タイムラインの概要) で、ライフサイクルルールを確認し、[Create rule] (ルールの作成) を選択します。

ライフサイクルルールの有効化

バケットのライフサイクルルールを有効または無効にするには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. ライフサイクルルールを有効または無効にする Outposts バケットを選択します。
4. [Management] (管理) タブを選択し、[Lifecycle rule] (ライフサイクルルール) で、有効または無効にするルールを選択します。
5. [Action] (アクション) で、[Enable or disable rule] (ルールを有効化または無効化) を選択します。

ライフサイクルルールの編集

1. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. ライフサイクルルールを編集する Outposts バケットを選択します。
4. [Management] (管理) タブを選択し、編集するライフサイクルルールを選択します。
5. (オプション) [Lifecycle rule name] (ライフサイクルルール名) の値を更新します。

6. [Rule scope] (ルールスコープ) で、必要に応じてスコープを編集します。

   - 特定のフィルターの適用範囲を制限するには、[Limit the scope of this rule using one or more filters] (1 つまたは複数のフィルターを使用してこのルールの適用範囲を制限) を選択します。次に、プレフィックスフィルターまたはオブジェクトサイズを追加します。
   - このルールをバケット内のすべてのオブジェクトに適用するには、[Apply to all objects in the bucket] (バケット内のすべてのオブジェクトに適用) を選択します。

7. [Lifecycle rule actions] (ライフサイクルルールのアクション) で、次のいずれかのオプションを選択します。

   - [Expire current versions of objects] (オブジェクトの現行バージョンを期限切れにする) — バージョニングが有効なバケットの場合、S3 on Outposts は削除マーカーを追加し、オブジェクトを最新でないバージョンとして保持します。S3 バージョニングを使用しないバケットの場合、S3 on Outposts はオブジェクトを完全に削除します。
   - [Permanently delete noncurrent versions of objects] (オブジェクトの最新ではないバージョンを完全に削除する) — S3 on Outposts は、オブジェクトの最新ではないバージョンを完全に削除します。
   - [Delete expired object delete markers or incomplete multipart uploads] (期限切れのオブジェクト削除マーカーまたは未完了のマルチパートアップロードを削除する) — S3 on Outposts は、期限切れのオブジェクト削除マーカーまたは未完了のマルチパートアップロードを完全に削除します。

     オブジェクトタグを使用してライフサイクルルールの範囲を制限する場合、[Delete expired object delete markers] (期限切れのオブジェクト削除マーカーを削除する) を選択することはできません。また、[Expire current object versions] (オブジェクトの現行バージョンを期限切れにする) を選択した場合は、[Delete expired object delete markers] (期限切れのオブジェクト削除マーカーを削除する) を選択することもできません。

   > **Note**
   >
   > サイズベースのフィルターは、削除マーカーや不完全なマルチパートアップロードには使用できません。

8. [Expire current versions of objects] (オブジェクトの現行バージョンを期限切れにする) または [Permanently delete noncurrent versions of objects] (オブジェクトの最新ではないバージョンを

完全に削除する) を選択した場合は、特定の日付またはオブジェクトの経過時間に基づいてルールトリガーを設定します。

9. [Delete expired object delete markers] (期限切れのオブジェクト削除マーカーを削除する) を選択した場合、期限切れのオブジェクト削除マーカーの削除を確認するには、[Delete expired object delete markers] (期限切れのオブジェクト削除マーカーを削除する) を選択します。
10. [Save (保存)] を選択します。

ライフサイクルルールの削除

1. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. ライフサイクルルールを削除する Outposts バケットを選択します。
4. [Management] (管理) タブを選択し、[Lifecycle rule] (ライフサイクルルール) で、削除するルールを選択します。
5. [Delete] (削除) をクリックします。

AWS CLI および SDK for Java を使用したライフサイクル設定の作成と管理

S3 ライフサイクルを使用して、Amazon S3 on Outposts のストレージ容量を最適化することができます。ライフサイクルルールを作成して、オブジェクトが古くなったり、新しいバージョンに置き換えられたりしたときに期限切れにすることができます。ライフサイクルルールを作成、有効化、無効化、または削除できます。

S3 ライフサイクルの詳細については、「ストレージのライフサイクルの管理」を参照してください。

> **Note**
>
> バケットを作成する AWS アカウント がそのバケットを所有し、ライフサイクルルールを作成、有効化、無効化、または削除できる唯一のアカウントです。

AWS Command Line Interface (AWS CLI) と AWS SDK for Java を使用して S3 on Outposts バケット用のライフサイクル設定を作成および管理するには、以下の例を参照してください。

トピック

- ライフサイクル設定を PUT する
- S3 on Outposts バケットのライフサイクル設定を GET する

ライフサイクル設定を PUT する

AWS CLI

次の AWS CLI の例では、Outposts バケットにライフサイクル設定ポリシーを配置します。このポリシーでは、フラグ付きのプレフィックス (*myprefix*) とタグを持つすべてのオブジェクトが 10 日後に期限切れになることを指定します。この例を実行するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。

1. ライフサイクル設定ポリシーを JSON ファイルに保存します。この例では、ファイル名は `lifecycle1.json` です。

```
{
    "Rules": [
        {
            "ID": "id-1",
            "Filter": {
                "And": {
                    "Prefix": "myprefix",
                    "Tags": [
                        {
                            "Value": "mytagvalue1",
                            "Key": "mytagkey1"
                        },
                        {
                            "Value": "mytagvalue2",
                            "Key": "mytagkey2"
                        }
                    ],
                    "ObjectSizeGreaterThan": 1000,
                    "ObjectSizeLessThan": 5000
                }
            },
            "Status": "Enabled",
            "Expiration": {
                "Days": 10
            }
        }
    ]
```

```
}
```

2. put-bucket-lifecycle-configuration CLI コマンドの一部として JSON ファイルを送信します。このコマンドを使用するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。このコマンドの詳細については、「AWS CLI リファレンス」の「put-bucket-lifecycle-configuration」を参照してください。

```
aws s3control put-bucket-lifecycle-configuration --account-id 123456789012 --
bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/
bucket/example-outposts-bucket --lifecycle-configuration file://lifecycle1.json
```

SDK for Java

次の SDK for Java の例では、Outposts バケットにライフサイクル設定を配置します。このライフサイクル設定では、フラグ付きのプレフィックス (*myprefix*) とタグを持つすべてのオブジェクトが 10 日後に期限切れになることを指定します。この例を実行するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。詳細については、「Amazon S3 API リファレンス」の「PutBucketLifecycleConfiguration」を参照してください。

```
import com.amazonaws.services.s3control.model.*;

public void putBucketLifecycleConfiguration(String bucketArn) {

    S3Tag tag1 = new S3Tag().withKey("mytagkey1").withValue("mytagkey1");
    S3Tag tag2 = new S3Tag().withKey("mytagkey2").withValue("mytagkey2");

    LifecycleRuleFilter lifecycleRuleFilter = new LifecycleRuleFilter()
            .withAnd(new LifecycleRuleAndOperator()
                    .withPrefix("myprefix")
                    .withTags(tag1, tag2))
                    .withObjectSizeGreaterThan(1000)
                    .withObjectSizeLessThan(5000);

    LifecycleExpiration lifecycleExpiration = new LifecycleExpiration()
            .withExpiredObjectDeleteMarker(false)
            .withDays(10);

    LifecycleRule lifecycleRule = new LifecycleRule()
            .withStatus("Enabled")
            .withFilter(lifecycleRuleFilter)
            .withExpiration(lifecycleExpiration)
```

```
            .withID("id-1");

    LifecycleConfiguration lifecycleConfiguration = new LifecycleConfiguration()
            .withRules(lifecycleRule);

    PutBucketLifecycleConfigurationRequest reqPutBucketLifecycle = new
 PutBucketLifecycleConfigurationRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withLifecycleConfiguration(lifecycleConfiguration);

    PutBucketLifecycleConfigurationResult respPutBucketLifecycle =
 s3ControlClient.putBucketLifecycleConfiguration(reqPutBucketLifecycle);
    System.out.printf("PutBucketLifecycleConfiguration Response: %s%n",
 respPutBucketLifecycle.toString());

}
```

S3 on Outposts バケットのライフサイクル設定を GET する

AWS CLI

次の AWS CLI の例では、Outposts バケットにライフサイクル設定を取得します。このコマンドを使用するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。このコマンドの詳細については、「AWS CLI リファレンス」の「put-bucket-lifecycle-configuration」を参照してください。

```
aws s3control get-bucket-lifecycle-configuration --account-id 123456789012 --bucket
 arn:aws:s3-outposts:<your-region>:123456789012:outpost/op-01ac5d28a6a232904/
bucket/example-outposts-bucket
```

SDK for Java

次の SDK for Java の例では、Outposts バケットにライフサイクル設定を取得します。詳細については、「Amazon Simple Storage Service API リファレンス」の「GetBucketLifecycleConfiguration」を参照してください。

```
import com.amazonaws.services.s3control.model.*;

public void getBucketLifecycleConfiguration(String bucketArn) {
```

```
    GetBucketLifecycleConfigurationRequest reqGetBucketLifecycle = new
 GetBucketLifecycleConfigurationRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    GetBucketLifecycleConfigurationResult respGetBucketLifecycle =
 s3ControlClient.getBucketLifecycleConfiguration(reqGetBucketLifecycle);
    System.out.printf("GetBucketLifecycleConfiguration Response: %s%n",
 respGetBucketLifecycle.toString());

}
```

S3 on Outposts のオブジェクトのレプリケート

AWS Outposts で S3 レプリケーションを有効にすると、Amazon S3 on Outposts を設定して、S3 オブジェクトを異なる Outposts 間、または同じ Outpost のバケット間で自動的にレプリケートできます。S3 Replication on Outposts を使用すると、同じまたは異なる Outposts、または異なるアカウント間でデータの複数のレプリカを管理できるため、データの常駐要件を満たすことができます。S3 Replication on Outposts は、準拠したストレージのニーズとアカウント間のデータ共有を強化するのに役立ちます。レプリカがソースデータと同一であることを確認する必要がある場合は、S3 Replication on Outposts を使用して、元のオブジェクトの作成時間、タグ、バージョン ID などのすべてのメタデータを保持するオブジェクトのレプリカを作成できます。

S3 Replication on Outposts では、バケット間のオブジェクトレプリケーションのステータスをモニタリングするための詳細なメトリクスと通知も提供されます。Amazon CloudWatch を使用して、レプリケーション保留中のバイト数、レプリケーション保留中のオペレーション、およびソースバケットとターゲットバケット間のレプリケーションレイテンシーを追跡することで、レプリケーションの進行状況を監視できます。設定の問題をすばやく診断して修正するには、レプリケーションオブジェクトの障害に関する通知を受け取るように Amazon EventBridge を設定することもできます。詳細については、「レプリケーションの管理」を参照してください。

トピック

- レプリケーション設定
- S3 Replication on Outposts の要件
- レプリケーションの対象
- レプリケーションの対象外
- S3 Replication on Outposts でサポートされていないものは何ですか?

- レプリケーションの設定
- レプリケーションの管理

レプリケーション設定

S3 on Outposts はレプリケーション設定を XML 形式で保存します。レプリケーション設定 XML ファイルで、AWS Identity and Access Management (IAM) ロールと 1 つ以上のルールを指定します。

```
<ReplicationConfiguration>
    <Role>IAM-role-ARN</Role>
    <Rule>
        ...
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
</ReplicationConfiguration>
```

S3 on Outposts はユーザーの許可なしにオブジェクトをレプリケートすることはできません。レプリケーション設定で指定した IAM ロールを使用して S3 on Outposts アクセス許可を付与します。S3 on Outposts は、ユーザーに代わってオブジェクトをレプリケートするための IAM ロールを引き受けます。レプリケーションを開始する前に、必要なアクセス許可を IAM ロールに付与する必要があります。S3 on Outposts に対するこれらのアクセス許可の詳細については、「IAM ロールの作成」を参照してください。

次のシナリオでは、レプリケーション設定にルールを 1 つ追加します。

- すべてのオブジェクトをレプリケートします。
- オブジェクトのサブセットをレプリケートします。ルールにフィルターを追加して、オブジェクトのサブセットを特定します。フィルターでは、ルールが適用されるオブジェクトのサブセットを特定するために、オブジェクトキープレフィックス、タグ、またはその両方の組み合わせを指定します。

オブジェクトの異なるサブセットをレプリケートする場合は、レプリケーション設定に複数のルールを追加します。各ルールでは、オブジェクトの異なるサブセットを選択するフィルターを指定します。例えば、`tax/` または `document/` のいずれかのキープレフィックスを持つオブジェクトをレ

プリケートするとします。このためには、tax/ キープレフィックスフィルターを指定するルールと、document/ キープレフィックスを指定するもう 1 つのルールの 2 つのルールを追加します。

S3 on Outposts のレプリケーション設定とレプリケーションルールの詳細については、Amazon Simple Storage Service API リファレンスの「ReplicationConfiguration」を参照してください。

S3 Replication on Outposts の要件

レプリケーションには以下が必要です。

- 送信先の Outpost CIDR 範囲は、ソースの Outpost サブネットテーブルに関連付けられている必要があります。詳細については、「レプリケーションルールの作成の前提条件」を参照してください。
- レプリケート元とレプリケート先の両方のバケットで、S3 バージョニングを有効にする必要があります。バージョニングの詳細については、S3 on Outposts バケットの S3 バージョニングの管理を参照してください。
- Amazon S3 on Outposts には、ユーザーに代わってレプリケート元バケットのオブジェクトをレプリケート先バケットにレプリケートするアクセス許可が必要です。つまり、GET および PUT のアクセス許可を S3 on Outposts に委任するには、サービスロールを作成する必要があります。
  1. サービスロールを作成する前に、レプリケート元バケットに対する GET アクセス許可とレプリケート先バケットに対する PUT アクセス許可が必要です。
  2. S3 on Outposts にアクセス許可を委任するサービスロールを作成するには、まず IAM エンティティ (ユーザーまたはロール) が iam:CreateRole および iam:PassRole アクションを実行できるようにアクセス許可を設定する必要があります。次に、IAM エンティティがサービスロールを作成できるようにします。S3 on Outposts がユーザーに代わってサービス ロールを引き受け、GET および PUT のアクセス許可を S3 on Outposts に委任するには、必要な信頼とアクセス許可のポリシーをロールに割り当てる必要があります。S3 on Outposts に対するこれらのアクセス許可の詳細については、「IAM ロールの作成」を参照してください。サービスロールの作成の詳細については、「サービスのロールの作成」を参照してください。

レプリケーションの対象

デフォルトで、S3 on Outposts は以下をレプリケートします。

- レプリケーション設定の追加後に作成されたオブジェクト。

- レプリケート元オブジェクトからレプリカへのオブジェクトメタデータ。レプリカからレプリケート元オブジェクトへのメタデータのレプリケーションについては、「Outposts の Amazon S3 レプリカ変更の同期が有効になっている場合のレプリケーションステータス」を参照してください。
- オブジェクトタグ、存在する場合。

削除オペレーションがレプリケーションに与える影響

レプリケート元バケットからオブジェクトを削除すると、デフォルトで次のアクションが実行されます。

- オブジェクトバージョン ID を指定せずに DELETE リクエストを行った場合、S3 on Outposts は削除マーカーを追加します。S3 on Outposts では、削除マーカーを次のように扱います。
  - S3 on Outposts は、デフォルトでは削除マーカーをレプリケートしません。
  - ただし、タグベース以外のルールには削除マーカーレプリケーションを追加できます。レプリケーション設定で削除マーカーレプリケーションを有効にする場合は、「S3 コンソールの使用」を参照してください。
- DELETE リクエストで削除するオブジェクトバージョン ID を指定した場合、S3 on Outposts はレプリケート元バケット内のそのオブジェクトバージョンを完全に削除します。しかし、レプリケート先バケット内でその削除をレプリケートすることはありません。これは、レプリケート先バケットからは、同じオブジェクトバージョンを削除しないことを意味します。これは悪意のある削除からデータを保護します。

レプリケーションの対象外

デフォルトで、S3 on Outposts は以下をレプリケートしません。

- 別のレプリケーションルールによって作成されたレプリカである、レプリケート元バケットのオブジェクト。例えば、バケット A がレプリケート元でバケット B がレプリケート先であるレプリケーションを設定するとします。ここで、バケット B をレプリケート元、バケット C をレプリケート先とする別のレプリケーション設定を追加したとします。この場合、バケット A のオブジェクトのレプリカであるバケット B のオブジェクトは、バケット C にレプリケートされません。
- 既に別のレプリケート先にレプリケートされている、レプリケート元バケット内のオブジェクト。例えば、既存のレプリケーション設定でレプリケート先バケットを変更した場合、S3 on Outposts がそのオブジェクトを再度レプリケートすることはありません。

- お客様が用意した暗号化キー (SSE-C) を使用し、サーバー側の暗号化で作成されたオブジェクト。
- バケットレベルのサブリソースの更新

  たとえば、ライフサイクル設定を変更したり、レプリケート元バケットに通知設定を追加した場合、これらの変更はレプリケート先バケットには適用されません。この機能により、レプリケート元バケットとレプリケート先バケットで異なる設定を指定できます。
- ライフサイクル設定によって実行されたアクション。

  例えば、ライフサイクル設定がレプリケート元バケットでのみ有効で、有効期限アクションを設定した場合、S3 on Outposts はレプリケート元バケットの有効期限切れになったオブジェクトの削除マーカーを作成しますが、それらのマーカーはレプリケート先バケットにレプリケートされません。同じライフサイクル設定をレプリケート元バケットとレプリケート先バケットの両方に適用する場合は、両方で同じライフサイクル設定を有効にします。ライフサイクル設定についての詳細は、ストレージのライフサイクルの管理 を参照してください。

S3 Replication on Outposts でサポートされていないものは何ですか?

S3 レプリケーションの以下の機能は、現時点では S3 on Outposts でサポートされていません。

- S3 Replication Time Control (S3 RTC) S3 Replication on Outposts のオブジェクトトラフィックはオンプレミスネットワーク (ローカルゲートウェイ) を経由するため、S3 RTC はサポートされていません。ローカルゲートウェイの詳細については、AWS Outposts ユーザーガイドの「ローカルゲートウェイの操作」を参照してください。
- バッチオペレーションの S3 レプリケーション

レプリケーションの設定

> **Note**
>
> レプリケーションルールをセットアップする前にバケットに存在していたオブジェクトは、自動的にレプリケートされません。つまり、Amazon S3 on Outposts はさかのぼってオブジェクトをレプリケートしません。レプリケーション設定の前に作成されたオブジェクトをレプリケートするには、`CopyObject` API オペレーションを使用して、それらを同じバケットにコピーします。オブジェクトがコピーされると、バケットに「新しい」オブジェクトとして表示され、レプリケーション設定が適用されます。オブジェクトのコピーの詳細については、Amazon Simple Storage Service API リファレンスの「AWS SDK for Java を使用して

Amazon S3 on Outposts バケットにオブジェクトをコピーする」および「CopyObject」を参照してください。

S3 Replication on Outposts を有効にするには、レプリケート元 Outposts バケットにレプリケーションルールを追加します。レプリケーションルールは、指定されたとおりにオブジェクトをレプリケートするように S3 on Outposts に指示します。レプリケーションルールでは、以下の項目を指定する必要があります。

- レプリケート元 Outposts バケットアクセスポイント – アクセスポイント Amazon Resource Name (ARN) または S3 on Outposts でオブジェクトをレプリケートするバケットのアクセスポイントエイリアス。アクセスポイントのエイリアスの使用の詳細については、「Using a bucket-style alias for your S3 on Outposts bucket access point」(S3 on Outposts バケットアクセスポイントにバケット形式のエイリアスを使用する) を参照してください。
- レプリケートするオブジェクト – レプリケート元 Outposts バケットまたはサブセット内のすべてのオブジェクトをレプリケートできます。サブセットを特定するには、キー名のプレフィックス、1 つ以上のオブジェクトタグ、またはその両方を設定で指定します。

  例えば、キー名のプレフィックス `Tax/` の オブジェクトのみをレプリケートするようにレプリケーションルールを設定した場合、S3 on Outposts は `Tax/doc1` や `Tax/doc2` などのキーを持つオブジェクトをレプリケートします。しかし、`Legal/doc3` というキーを持つオブジェクトはレプリケートしません。プレフィックスと 1 つ以上のタグの両方を指定した場合、S3 on Outposts は特定のキープレフィックスとタグを持つオブジェクトのみをレプリケートします。
- レプリケート先の Outposts バケット — S3 on Outposts でオブジェクトをレプリケートするバケットの ARN またはアクセスポイントエイリアス。

REST API、AWS SDK、AWS Command Line Interface (AWS CLI)、または Amazon S3 コンソールを使用してレプリケーションルールを設定できます。

S3 on Outposts は、レプリケーションルールの設定をサポートする API も提供します。詳細については、Amazon Simple Storage Service API リファレンスの次のトピックを参照してください。

- PutBucketReplication
- GetBucketReplication
- DeleteBucketReplication

トピック

- レプリケーションルールの作成の前提条件
- Outposts でのレプリケーションルールの作成

レプリケーションルールの作成の前提条件

トピック

- レプリケート元とレプリケート先の Outpost サブネットの接続
- IAM ロールの作成

レプリケート元とレプリケート先の Outpost サブネットの接続

レプリケーショントラフィックをレプリケート元 Outpost からレプリケート先 Outpost にローカルゲートウェイ経由で送信するには、新しいルートを追加してネットワークを設定する必要があります。アクセスポイントの Classless Inter-Domain Routing (CIDR) ネットワーク範囲を相互に接続する必要があります。アクセスポイントのペアごとに、この接続を一度だけセットアップする必要があります。

接続を設定する手順は、アクセスポイントに関連付けられている Outposts エンドポイントのアクセスタイプによって異なります。エンドポイントのアクセスタイプは、[プライベート] (AWS Outposts のルーティング用直接仮想プライベートクラウド (VPC)) または [カスタマー所有の IP] (オンプレミスネットワーク内のカスタマー所有の IP アドレスプール [CoIP プール]) のいずれかです。

ステップ 1: レプリケート元 Outposts エンドポイントの CIDR 範囲を確認する

レプリケート元アクセスポイントに関連付けられているレプリケート元エンドポイントの CIDR 範囲を確認するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. [Outposts バケット] リストで、レプリケーションに必要なレプリケート元バケットを選択します。
4. [Outposts アクセスポイント] タブを選択し、レプリケーションルールのレプリケート元バケットの Outposts アクセスポイントを選択します。
5. Outposts エンドポイントを選択します。
6. ステップ 5 で使用するサブネット ID をコピーします。

7. レプリケート元 Outposts エンドポイントの CIDR 範囲を見つけるために使用する方法は、エンドポイントのアクセスタイプによって異なります。

   Outposts エンドポイントの概要セクションで、「アクセスタイプ」を参照してください。

   - アクセスタイプが [プライベート] の場合は、Classless Inter-Domain Routing (CIDR) 値をコピーしてステップ 6 で使用します。
   - アクセスタイプが [カスタマー所有の IP] の場合は、次の操作を行います。
     1. お客様所有の IPv4 プール値をコピーして、後でアドレスプールの ID として使用します。
     2. AWS Outposts コンソール (https://console.aws.amazon.com/outposts/) を開きます。
     3. ナビゲーションペインで [ローカルゲートウェイのルートテーブル] をクリックします。
     4. レプリケート元 Outpost のローカルゲートウェイルートテーブル ID 値を選択します。
     5. 詳細ペインの [CoIP プール] タブを選択します。以前にコピーした CoIP プール ID の値を検索ボックスに貼り付けます。
     6. 一致した CoIP プールについては、ステップ 6 で使用するレプリケート元 Outposts エンドポイントの対応する CIDR 値をコピーします。

ステップ 2: レプリケート先の Outposts エンドポイントのサブネット ID と CIDR 範囲を確認する

レプリケート先アクセスポイントに関連付けられているレプリケート先エンドポイントのサブネット ID と CIDR 範囲を確認するには、ステップ 1 と同じ手順に従い、それらのサブステップを適用するときに、レプリケート元 Outposts エンドポイントをレプリケート先 Outposts エンドポイントに変更します。ステップ 6 で使用するために、レプリケート先 Outposts エンドポイントのサブネット ID 値をコピーします。ステップ 5 で使用するために、レプリケート先 Outposts エンドポイントの CIDR 値をコピーします。

ステップ 3: レプリケート元 Outpost のローカルゲートウェイ ID を確認する

レプリケート元 Outpost のローカルゲートウェイ ID を確認するには

1. AWS Outposts コンソール (https://console.aws.amazon.com/outposts/) を開きます。
2. 左のナビゲーションペインで [ローカルゲートウェイ] を選択します。
3. [ローカルゲートウェイ] ページで、レプリケーションに使用するレプリケート元 Outpost ID を確認します。
4. ステップ 5 で使用するために、レプリケート元 Outpost のローカルゲートウェイ ID 値をコピーします。

ローカルゲートウェイの詳細については、AWS Outposts ユーザーガイドの「Local gateway」(ローカルゲートウェイ) を参照してください。

ステップ 4: レプリケート先 Outpost のローカルゲートウェイ ID を確認する

レプリケート先 Outpost のローカルゲートウェイ ID を確認するには、レプリケート先の Outpost ID を検索する点を除き、ステップ 3 と同じサブステップに従います。ステップ 6 で使用するために、レプリケート先 Outpost のローカルゲートウェイ ID 値をコピーします。

ステップ 5: レプリケート元 Outpost サブネットからレプリケート先 Outpost サブネットへの接続をセットアップする

レプリケート元 Outpost サブネットからレプリケート先 Outpost サブネットに接続するには

1. AWS Management Console にサインインして、VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. 左側のナビゲーションペインで [Subnets] (サブネット) を選択します。
3. 検索ボックスに、ステップ 1 で確認したレプリケート元 Outposts エンドポイントのサブネット ID を入力します。一致するサブネット ID のサブネットを選択します。
4. 一致するサブネット項目には、このサブネットのルートテーブル値を選択します。
5. ルートテーブルが選択されたページで、[アクション] を選択し、次に [ルートの編集] を選択します。
6. [ルートの編集] タブで、[ルートの追加] を選択します。
7. [送信先] に、ステップ 2 で確認したレプリケート先 Outposts エンドポイントの CIDR 範囲を入力します。
8. [ターゲット] で [Outpost Local Gateway] を選択し、ステップ 3 で見つけたレプリケート元 Outpost のローカルゲートウェイ ID を入力します。
9. [Save changes] (変更の保存) をクリックします。
10. ルートのステータスが [アクティブ] であることを確認します。

ステップ 6: レプリケート先 Outpost サブネットからレプリケート元 Outpost サブネットへの接続をセットアップする

1. AWS Management Console にサインインして、VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. 左側のナビゲーションペインで [Subnets] (サブネット) を選択します。

3. 検索ボックスに、ステップ 2 で確認したレプリケート先 Outposts エンドポイントのサブネット ID を入力します。一致するサブネット ID のサブネットを選択します。
4. 一致するサブネット項目には、このサブネットのルートテーブル値を選択します。
5. ルートテーブルが選択されたページで、[アクション] を選択し、次に [ルートの編集] を選択します。
6. [ルートの編集] タブで、[ルートの追加] を選択します。
7. [送信先] に、ステップ 1 で確認したレプリケート元 Outposts エンドポイントの CIDR 範囲を入力します。
8. [ターゲット] で [Outpost Local Gateway] を選択し、ステップ 4 で見つけたレプリケート先 Outpost のローカルゲートウェイ ID を入力します。
9. [Save changes] (変更の保存) をクリックします。
10. ルートのステータスが [アクティブ] であることを確認します。

レプリケート元とレプリケート先のアクセスポイントの CIDR ネットワーク範囲を接続したら、AWS Identity and Access Management (IAM) ロールを作成する必要があります。

IAM ロールの作成

デフォルトで、すべての S3 on Outposts リソース (バケット、オブジェクト、関連するサブリソース) はプライベートであり、リソース所有者のみがリソースにアクセスできます。S3 on Outposts には、ソース Outposts バケットからオブジェクトを読み取って、レプリケートするアクセス許可が必要です。IAM サービスロールを作成してこれらの許可を付与し、その後、レプリケーション設定でそのロールを指定します。

このセクションでは、信頼ポリシーと最低限必要なアクセス許可ポリシーについて説明します。チュートリアル例では、IAM ロールを作成するための手順をステップバイステップで説明しています。詳細については、「Outposts でのレプリケーションルールの作成」を参照してください。IAM ロールの詳細については、IAM ユーザーガイドの IAM ロールを参照してください。

- 以下の例は、信頼ポリシーを示しています。ここでは、このロールを引き受けることができるサービスプリンシパルとして S3 on Outposts を特定します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
```

```
         "Principal":{
            "Service":"s3-outposts.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

- 以下の例は、アクセスポリシーを示しています。ここでは、ユーザーに代わってレプリケーションタスクを実行する権許可をロールに付与します。S3 on Outposts がこのロールを引き受ける場合、このポリシーで指定されたアクセス許可を持つことになります。このポリシーを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。レプリケート元とレプリケート先 Outposts の Outposts ID およびレプリケート元とレプリケート先 Outposts バケットのバケット名とアクセスポイント名に置き換えてください。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:GetObjectVersionForReplication",
            "s3-outposts:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
bucket/SOURCE-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
accesspoint/SOURCE-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:ReplicateObject",
            "s3-outposts:ReplicateDelete"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/
bucket/DESTINATION-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/
accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
```

```
      }
   ]
}
```

アクセスポリシーは、以下のアクションに対するアクセス許可を付与します。

- `s3-outposts:GetObjectVersionForReplication` — このアクションのためのアクセス許可はすべてのオブジェクトに付与され、各オブジェクトに関連する特定のオブジェクトバージョンを S3 on Outposts が取得できるようにします。

- `s3-outposts:GetObjectVersionTagging` — *SOURCE-OUTPOSTS-BUCKET* バケット (レプリケート元バケット) のオブジェクトに対するこのアクションのアクセス許可により、S3 on Outposts はレプリケーションのためにオブジェクトタグを読み取ることができるようになります。詳細については、「S3 on Outposts バケットのタグの追加」を参照してください。S3 on Outposts にこのアクセス許可がない場合は、オブジェクトはレプリケートされますが、オブジェクトタグはレプリケートされません。

- `s3-outposts:ReplicateObject` および `s3-outposts:ReplicateDelete` — *DESTINATION-OUTPOSTS-BUCKET* バケットのオブジェクトに属すこれらのアクションの許可により、S3 on Outposts はレプリケート先 Outposts バケットにオブジェクトまたは削除マーカーをレプリケートできます。削除マーカーの詳細については、「削除オペレーションがレプリケーションに与える影響」を参照してください。

  > **Note**
  >
  > - *DESTINATION-OUTPOSTS-BUCKET* バケット (レプリケート先バケット) に対する `s3-outposts:ReplicateObject` アクションのアクセス権限により、オブジェクトタグのレプリケーションも許可されます。したがって、`s3-outposts:ReplicateTags` アクションのアクセス許可を明示的に付与する必要はありません。
  > - クロスアカウントレプリケーションの場合、レプリケート先 Outposts バケットの所有者がバケットポリシーを更新して、*DESTINATION-OUTPOSTS-BUCKET* での `s3-outposts:ReplicateObject` アクションに対するアクセス許可を付与する必要があります。`s3-outposts:ReplicateObject` アクションにより、S3 on Outposts はレプリケート先 Outposts バケットにオブジェクトおよびオブジェクトタグをレプリケートできます。

S3 on Outposts アクションのリストについては、「S3 on Outposts によって定義されたアクション」を参照してください。

> **Important**
>
> IAM ロールを所有する AWS アカウントは、IAM ロールに付与するアクションの許可を持っている必要があります。
> 例えば、レプリケート元 Outposts バケットに別の AWS アカウントが所有するオブジェクトが含まれていたとします。オブジェクトの所有者は、IAM ロールを所有する AWS アカウントに、バケットポリシーとアクセスポイントポリシーを介して必要なアクセス許可を明示的に付与する必要があります。そうでない場合、S3 on Outposts はオブジェクトにアクセスできず、オブジェクトのレプリケーションは失敗します。
> ここで説明されているアクセス許可は、最小のレプリケーション設定に関連しています。オプションのレプリケーション設定を追加する場合は、追加のアクセス許可を S3 on Outposts に付与する必要があります。

レプリケーション元とレプリケーション先の Outposts バケットが異なる AWS アカウントによって所有されている場合のアクセス許可の付与

レプリケーション元とレプリケーション先の Outposts バケットが同じアカウントによって所有されていない場合、レプリケート先バケットの所有者は、バケットとレプリケート先バケットのアクセスポイントポリシーを更新する必要があります。これらのポリシーでは、次のポリシー例に示すように、レプリケート元 Outposts バケットの所有者と IAM サービスロールにレプリケーションアクションを実行するアクセス許可を付与する必要があり、そうしないと、レプリケーションは失敗します。このポリシーの例では、*DESTINATION-OUTPOSTS-BUCKET* はレプリケーション先バケットです。これらのポリシーの例を実行するには、*user input placeholders* をユーザー自身の情報に置き換えます。

IAM サービスロールを手動で作成する場合は、次のポリシー例に示すように、ロールパスを `role/service-role/` として設定します。詳細については、「IAM ユーザーガイド」の「IAM ARN」を参照してください。

```
{
   "Version":"2012-10-17",
   "Id":"PolicyForDestinationBucket",
   "Statement":[
      {
```

```
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3-outposts:ReplicateDelete",
            "s3-outposts:ReplicateObject"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:DestinationBucket-account-
ID:outpost/DESTINATION-OUTPOST-ID/bucket/DESTINATION-OUTPOSTS-BUCKET/object/*"
         ]
      }

   ]
}
```

```
{
"Version":"2012-10-17",
   "Id":"PolicyForDestinationAccessPoint",
   "Statement":[
      {
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3-outposts:ReplicateDelete",
            "s3-outposts:ReplicateObject"
         ],
         "Resource" :[
            "arn:aws:s3-outposts:region:DestinationBucket-account-
ID:outpost/DESTINATION-OUTPOST-ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/
object/*"
         ]
      }
   ]
}
```

> **Note**
>
> レプリケート元 Outposts バケットのオブジェクトがタグ付きの場合は、以下の点に注意してください。
> レプリケート元の Outposts バケット所有者が (IAM ロールによって) オブジェクトタグをレプリケートするための `s3-outposts:GetObjectVersionTagging` および `s3-outposts:ReplicateTags` アクションのアクセス許可を S3 on Outposts に付与した場合、Amazon S3 はオブジェクトと共にタグをレプリケートします。IAM ロールに関する詳細は、「IAM ロールの作成」を参照してください。

Outposts でのレプリケーションルールの作成

S3 Replication on Outposts は、同一または異なる AWS Outposts にあるバケット間でのオブジェクトの自動的で非同期のレプリケーションです。レプリケーションでは、新しく作成されたオブジェクトおよびオブジェクトの更新が、レプリケート元 Outposts バケットからレプリケート先 Outposts バケットにコピーされます。詳細については、「S3 on Outposts のオブジェクトのレプリケート」を参照してください。

> **Note**
>
> レプリケーションルールをセットアップする前にレプリケート元 Outposts バケットに存在したオブジェクト。つまり、S3 on Outposts はさかのぼってオブジェクトをレプリケートしません。レプリケーション設定の前に作成されたオブジェクトをレプリケートするには、`CopyObject` API オペレーションを使用して、それらを同じバケットにコピーします。オブジェクトがコピーされると、バケットに「新しい」オブジェクトとして表示され、レプリケーション設定が適用されます。オブジェクトのコピーの詳細については、Amazon Simple Storage Service API リファレンスの「AWS SDK for Java を使用して Amazon S3 on Outposts バケットにオブジェクトをコピーする」および「CopyObject」を参照してください。

レプリケーションを設定するときは、レプリケート元 Outposts バケットにレプリケーションルールを追加します。レプリケーションルールにより、レプリケート元 Outposts バケットオブジェクトと、レプリケートされたオブジェクトが保存されるレプリケート先 Outposts バケットが定義されます。ルールを作成して、バケット内のすべてのオブジェクト、または特定のキー名のプレフィックス、1 つ以上のオブジェクトタグ、あるいはその両方を持つオブジェクトのサブセットをレプリ

ケートできます。レプリケート先 Outposts バケットはレプリケート元 Outposts バケットと同じ Outposts にあっても、別の Outposts にあってもかまいません。

S3 on Outposts レプリケーションルールでは、レプリケート元とレプリケート先の バケット名の代わりに、レプリケート元 Outposts バケットのアクセスポイント Amazon リソースネーム (ARN) とレプリケート先の Outposts バケットのアクセスポイント ARN の両方を指定する必要があります。

削除するオブジェクトバージョンの ID を指定した場合、S3 on Outposts はレプリケート元 Outposts バケット内のそのオブジェクトバージョンを削除します。しかし、レプリケート先 Outposts バケット内でその削除をレプリケートすることはありません。つまり、レプリケート先 Outposts バケットから同じオブジェクトバージョンを削除しません。この動作は悪意のある削除からデータを保護します。

Outposts バケットにレプリケーションルールを追加すると、そのルールはデフォルトで有効になるため、保存するとすぐに動作を開始します。

この例では、異なる Outposts にあり、同じ AWS アカウントが所有する、レプリケート元 Outposts とレプリケート先 Outposts のバケットのレプリケーションのセットアップを行います。Amazon S3 コンソールを使用する例については、AWS Command Line Interface (AWS CLI)、AWS SDK for Java、および AWS SDK for .NET を参照してください。クロスアカウントの S3 Replication on Outposts アクセス許可の詳細については、「レプリケーション元とレプリケーション先の Outposts バケットが異なる AWS アカウントによって所有されている場合のアクセス許可の付与」を参照してください。

S3 on Outposts レプリケーションルールを設定するための前提条件については、「レプリケーションルールの作成の前提条件」を参照してください。

S3 コンソールの使用

以下の手順を実行して、レプリケート元 Outposts バケットと異なる Outposts にレプリケート先 Amazon S3 on Outposts バケットがある場合のレプリケーションルールを設定します。

レプリケート先 Outposts バケットがレプリケート元 Outposts バケットとは別のアカウントにある場合は、レプリケート先 Outposts バケットにバケットポリシーを追加して、レプリケート元 Outposts バケットアカウントの所有者に、レプリケート先 Outposts バケットのオブジェクトをレプリケートするアクセス許可を付与する必要があります。詳細については、「レプリケーション元とレプリケーション先のバケットが異なる AWS アカウントによって所有されている場合の許可の付与」を参照してください。

レプリケーションルールを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. [Outposts バケット] リストで、レプリケート元バケットとして使用するバケットの名前を選択します。
3. [管理] タブを選択し、[レプリケーションルール] までスクロールして、[レプリケーションルールを作成] を選択します。
4. [レプリケーションルール名] で、後でルールを識別しやすいようにルールの名前を入力します。ルール名は必須であり、バケット内で一意である必要があります。
5. [ステータス] では、デフォルトで [有効] が選択されています。有効にされたルールは、保存するとすぐに機能し始めます。ルールを後から有効にする場合は、[無効] を選択します。
6. [優先度] では、ルールの優先度値によって、重複するルールがある場合に適用するルールが決まります。オブジェクトが複数のレプリケーションルールの範囲に含まれる場合、S3 on Outposts はこれらの優先度の値を使用して競合を回避します。デフォルトでは、新しいルールが最優先でレプリケーション設定に追加されます。数値が大きいほど、優先度が高くなります。

   ルールの優先度を変更するには、ルールを保存した後、レプリケーションルールリストからルール名を選択し、[アクション] を選択して、[優先度の編集] を選択します。
7. [ソースバケット] には、レプリケーションソースを設定するための次のオプションがあります。

   - バケット全体をレプリケートするには、[バケット内のすべてのオブジェクトに適用] を選択します。
   - プレフィックスまたはタグフィルタリングをレプリケーションソースに適用するには、[1 つまたは複数のフィルターを使用してこのルールの適用範囲を制限します] を選択します。プレフィックスとタグを組み合わせることができます。
     - 同じプレフィックスを持つすべてのオブジェクトを複製するには、[プレフィックス] で、ボックスにプレフィックスを入力します。[プレフィックス] フィルターを使用すると、同じ文字列で始まる名前 (`pictures` など) を持つすべてのオブジェクトへのレプリケーションを制限します。

       フォルダの名前をプレフィックスとして入力する場合は、最後の文字に / (スラッシュ) を使用する必要があります (例: `pictures/`)。
     - 1 つまたは複数のオブジェクトタグを持つすべてのオブジェクトをレプリケートするには、[タグを追加] を選択し、ボックスにキーと値のペアを入力します。別のタグを追加する

には、この手順を繰り返します。オブジェクトロックの詳細については、S3 on Outposts バケットのタグの追加を参照してください。

8. S3 on Outposts のレプリケート元バケットにアクセスしてレプリケーションを行うには、[ソースアクセスポイント名] で、レプリケート元バケットに接続されているアクセスポイントを選択します。
9. [送信先] で、S3 on Outposts でオブジェクトをレプリケートするレプリケート先 Outposts バケットのアクセスポイント ARN を選択します。レプリケート先 Outposts バケットは、レプリケート元 Outposts バケットと同じまたは異なる AWS アカウント に配置することができます。

   レプリケート先バケットがレプリケート元 Outposts バケットとは別のアカウントにある場合は、レプリケート先 Outposts バケットにバケットポリシーを追加して、レプリケート元 Outposts バケットアカウントの所有者に、レプリケート先 Outposts バケットのオブジェクトをレプリケートするアクセス許可を付与する必要があります。詳細については、「レプリケーション元とレプリケーション先の Outposts バケットが異なる AWS アカウントによって所有されている場合のアクセス許可の付与」を参照してください。

   > **Note**
   >
   > レプリケート先 Outposts バケットでバージョニングが有効になっていない場合は、[バージョニングの有効化] ボタンを含む警告が表示されます。バケットでバージョニングを有効にするには、このボタンを選択します。

10. S3 on Outposts がユーザーのためにオブジェクトをレプリケートできる AWS Identity and Access Management (IAM) サービスロールを設定します。

    IAM ロールをセットアップするには、[IAM ロール] で、次のいずれかの操作を行います。

    - S3 on Outposts でレプリケーション設定用の新しい IAM ロールを作成するには、[既存の IAM ロールから選択する] を選択し、[新しいロールの作成] を選択します。ルールを保存すると、選択したレプリケート元 Outposts バケットとレプリケート先 Outposts バケットに一致する IAM ロールに対して新しいポリシーが生成されます。[新しいロールを作成] を選択することを推奨します。
    - 既存の IAM ロールの使用も選択できます。その場合は、レプリケーションに必要なアクセス許可を S3 on Outposts に付与するロールを選択する必要があります。このロールによってレプリケーションルールに従うための十分なアクセス許可が S3 on Outposts に付与されない場合、レプリケーションは失敗します。

既存のロールを選択するには、[既存の IAM ロールから選択] を選択してから、ドロップダウンメニューでロールを選択します。[IAM ロール ARN を入力する] を選択してから、IAM ロールの Amazon リソースネーム (ARN) を入力することもできます。

> **Important**
>
> S3 on Outposts バケットにレプリケーションルールを追加する場合は、S3 on Outposts レプリケーションアクセス許可を付与する IAM ロールを作成して渡すことができる `iam:CreateRole` と `iam:PassRole` アクセス許可を持つ必要があります。詳細については、IAM ユーザーガイドの「AWS のサービス のサービスにロールを渡すアクセス権限をユーザーに付与する」を参照してください。

11. Outposts バケットのすべてのオブジェクトは、デフォルトで暗号化されます。S3 on Outposts の暗号化の詳細については、「S3 on Outposts のデータ暗号化」を参照してください。Amazon S3 マネージドキー (SSE-S3) によるサーバー側の暗号化を使用して暗号化されたオブジェクトのみがレプリケートできます。AWS Key Management Service(AWS KMS) キーを使用したサーバー側の暗号化 (SSE-KMS) またはお客様が用意した暗号化キーを使用したサーバー側の暗号化 (SSE-C) を使用して暗号化されたオブジェクトのレプリケーションはサポートされていません。
12. 必要に応じて、レプリケーションルール設定を設定する際は、次の追加オプションを有効にします。

    - レプリケーション設定で S3 on Outposts レプリケーションメトリクスを有効にするには、[レプリケーションメトリクス] を選択します。詳細については、「レプリケーションメトリクスによる進行状況のモニタリング」を参照してください。
    - レプリケーション設定で削除マーカーレプリケーションを有効にする場合は、[削除マーカーのレプリケーション] を選択します。詳細については、「削除オペレーションがレプリケーションに与える影響」を参照してください。
    - レプリカに加えられたメタデータの変更をレプリケート元オブジェクトに複製する場合は、[レプリカ変更の同期] を選択します。詳細については、「Outposts の Amazon S3 レプリカ変更の同期が有効になっている場合のレプリケーションステータス」を参照してください。
13. [ロールを作成] を選択して終了します。

ルールを保存したら、ルールを編集、有効化、無効化、または削除できます。そのためには、レプリケート元 Outposts バケットの [管理] タブに移動し、[レプリケーションルール] セクションまでスクロールして、ルールを選択し、[ルールの編集] を選択します。

AWS CLI を使用する場合

レプリケート元 Outposts バケットとレプリケート先 Outposts バケットが同じ AWS アカウント によって所有されている場合に、レプリケーションの設定に AWS CLI を使用するには

- レプリケート元 Outposts バケットとレプリケート先 Outposts バケットを作成する
- 両方のバケットでバージョニングを有効化する
- オブジェクトをレプリケートするための S3 on Outposts 許可を付与する IAM ロールを作成します。
- レプリケート元 Outposts バケットにレプリケーション設定を追加します。

設定を確認するには、テストします。

レプリケート元 Outposts バケットとレプリケート先 Outposts バケットが同じ AWS アカウントによって所有されている場合にレプリケーションをセットアップするには

1. AWS CLI の認証情報プロファイルを設定します。この例では、プロファイル名 `acctA` を使用します。認証情報プロファイルの設定については、AWS Command Line Interface ユーザーガイドの「名前付きプロファイル」を参照してください。

   > **Important**
   >
   > この演習に使用するプロファイルは、必要なアクセス権限を持っている必要があります。例えば、レプリケーション設定で、S3 on Outposts が引き受けることができる IAM ロールを指定します。使用するプロファイルに `iam:CreateRole` および `iam:PassRole` アクセス許可がある場合のみ実行できます。詳細については、IAM ユーザーガイドの「AWS のサービス のサービスにロールを渡すアクセス権限をユーザーに付与する」を参照してください。管理者の認証情報を使用して名前付きプロファイルを作成すると、指定されたプロファイルにはすべてのタスクを実行するために必要なアクセス許可が付与されます。

2. *#######*バケットを作成してバージョニングを有効にします。次の `create-bucket` コマンドは、米国東部 (バージニア北部) (`us-east-1`) リージョンに *`SOURCE-OUTPOSTS-BUCKET`* バケットを作成します。このコマンドを使用するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
aws s3control create-bucket --bucket SOURCE-OUTPOSTS-BUCKET --outpost-id SOURCE-OUTPOST-ID --profile acctA --region us-east-1
```

次の put-bucket-versioning コマンドは、*SOURCE-OUTPOSTS-BUCKET* バケットでバージョニングを有効にします。このコマンドを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/bucket/SOURCE-OUTPOSTS-BUCKET --versioning-configuration Status=Enabled --profile acctA
```

3. *#######*バケットを作成してバージョニングを有効にします。次の create-bucket コマンドは、米国西部 (オレゴン) (us-west-2) リージョンに *DESTINATION-OUTPOSTS-BUCKET* バケットを作成します。このコマンドを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

> **Note**
>
> レプリケーション元とレプリケーション先の Outposts バケットの両方が同じ AWS アカウント にある場合、レプリケーション設定をセットアップするには、同じ名前付きプロファイルを使用します。この例では acctA を使用します。異なる AWS アカウント によってバケットが所有されている場合、レプリケーション設定をテストするには、それぞれのバケットに異なるプロファイルを指定します。

```
aws s3control create-bucket --bucket DESTINATION-OUTPOSTS-BUCKET --create-bucket-configuration LocationConstraint=us-west-2 --outpost-id DESTINATION-OUTPOST-ID --profile acctA --region us-west-2
```

次の put-bucket-versioning コマンドは、*DESTINATION-OUTPOSTS-BUCKET* バケットでバージョニングを有効にします。このコマンドを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/bucket/DESTINATION-OUTPOSTS-BUCKET --versioning-configuration Status=Enabled --profile acctA
```

4. [IAM サービスロールを作成します]。レプリケーション設定では、後で *SOURCE-OUTPOSTS-BUCKET* バケットにこのサービスロールを追加します。S3 on Outposts は、ユーザーに代わってオブジェクトをレプリケートするこのロールを引き受けます。IAM ロールは 2 つのステップで作成します。

   a. IAM ロールを作成します。

      i. 次の信頼ポリシーをコピーして、ローカルコンピュータの現在のディレクトリにある `s3-on-outposts-role-trust-policy.json` という名前のファイルに保存します。このポリシーは、サービスロールを引き受けるアクセス許可を S3 on Outposts サービスプリンシパルに付与します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3-outposts.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

      ii. 次のコマンドを実行してロールを作成します。*user input placeholders* を、ユーザー自身の情報に置き換えます。

```
aws iam create-role --role-name replicationRole --assume-role-policy-
document file://s3-on-outposts-role-trust-policy.json --profile acctA
```

   b. アクセス許可ポリシーをサービスロールにアタッチします。

      i. 次のアクセス権限ポリシーをコピーして、ローカルコンピュータの現在のディレクトリにある `s3-on-outposts-role-permissions-policy.json` という名前のファイルに保存します。このポリシーは、さまざまな S3 on Outposts バケットとオブジェクトアクションに対するアクセス許可を付与します。このポリシーを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
   "Version":"2012-10-17",
```

```
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:GetObjectVersionForReplication",
            "s3-outposts:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-
OUTPOST-ID/bucket/SOURCE-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-
OUTPOST-ID/accesspoint/SOURCE-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:ReplicateObject",
            "s3-outposts:ReplicateDelete"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-
OUTPOST-ID/bucket/DESTINATION-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-
OUTPOST-ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      }
   ]
}
```

ii. ポリシーを作成してロールにアタッチするには、次のコマンドを実行します。*user input placeholders* を、ユーザー自身の情報に置き換えます。

```
aws iam put-role-policy --role-name replicationRole --policy-
document file://s3-on-outposts-role-permissions-policy.json --policy-
name replicationRolePolicy --profile acctA
```

5. *SOURCE-OUTPOSTS-BUCKET* バケットにレプリケーション設定を追加します。

   a. S3 on Outposts API は XML 形式でのレプリケーション設定を要求しますが、AWS CLI ではレプリケーション設定を JSON 形式で指定する必要があります。以下の JSON を、コンピュータのローカルディレクトリの `replication.json` というファイルに保存します。

この設定を使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
  "Role": "IAM-role-ARN",
  "Rules": [
    {
      "Status": "Enabled",
      "Priority": 1,
      "DeleteMarkerReplication": { "Status": "Disabled" },
      "Filter" : { "Prefix": "Tax"},
      "Destination": {
        "Bucket":
        "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-
ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT"
      }
    }
  ]
}
```

b. 次の put-bucket-replication コマンドを実行して、レプリケート元 Outposts バケットにレプリケーション設定を追加します。このコマンドを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control put-bucket-replication --account-id 123456789012 --
bucket arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-
ID/bucket/SOURCE-OUTPOSTS-BUCKET --replication-configuration file://
replication.json --profile acctA
```

c. レプリケーション設定を取得するには、get-bucket-replication コマンドを使用します。このコマンドを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
aws s3control get-bucket-replication --account-id 123456789012 --bucket
 arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
bucket/SOURCE-OUTPOSTS-BUCKET --profile acctA
```

6. Amazon S3 コンソールでセットアップをテストします。

a. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。

b. *SOURCE-OUTPOSTS-BUCKET* バケットに Tax という名前のフォルダを作成します。

c. *SOURCE-OUTPOSTS-BUCKET* バケット内の Tax フォルダにサンプルオブジェクトを追加します。

d. *DESTINATION-OUTPOSTS-BUCKET* バケットで、以下の点を確認します。

- S3 on Outposts がオブジェクトをレプリケートしました。

> **Note**
>
> S3 on Outposts がオブジェクトをレプリケートするのにかかる時間は、オブジェクトのサイズによって異なります。レプリケーションのステータスを確認する方法については、「レプリケーションステータス情報の取得」を参照してください。

- オブジェクトの [プロパティ] タブで、[レプリケーションステータス] が [レプリカ] に設定されています (これをレプリカオブジェクトとして識別)。

レプリケーションの管理

このセクションでは、S3 on Outposts で使用可能な追加のレプリケーション設定オプション、レプリケーションステータスの確認方法、レプリケーションのトラブルシューティング方法について説明します 主要なレプリケーション設定については、「レプリケーションの設定」を参照してください。

トピック

- レプリケーションメトリクスによる進行状況のモニタリング
- レプリケーションステータス情報の取得
- レプリケーションのトラブルシューティング
- S3 Replication on Outposts に EventBridge を使用する

レプリケーションメトリクスによる進行状況のモニタリング

S3 Replication on Outposts は、レプリケーション設定のレプリケーションルールの詳細なメトリクスを提供します。レプリケーションメトリクスを使用してレプリケーションの進行状況を 5 分間隔でモニタリングするには、保留中のレプリケーションのバイト数、レプリケーションのレイテンシー、レプリケーション、保留中のレプリケーションの操作を追跡します。設定の問題のトラブルシュー

ティングを支援するために、Amazon EventBridge を設定して、レプリケーションの失敗に関する通知を受け取ることもできます。

レプリケーションメトリクスを有効にすると、S3 Replication on Outposts は次のメトリクスを Amazon CloudWatch に発行します。

- レプリケーションを保留しているバイト – 特定のレプリケーションルールのレプリケーションを保留しているオブジェクトの合計バイト数。
- レプリケーションレイテンシー – 特定のレプリケーションルールで、レプリケート先バケットがレプリケート元バケットの背後にある最大秒数。
- レプリケーションを保留している操作 – 特定のレプリケーションルールでレプリケーションを保留している操作の数。操作には、オブジェクト、削除マーカー、タグが含まれます

> **Note**
>
> S3 Replication on Outposts は、CloudWatch カスタムメトリクスと同じ料金レートで請求されます。詳細については、「CloudWatch 料金表」を参照してください。

レプリケーションステータス情報の取得

レプリケーションステータスは、Amazon S3 on Outposts によってレプリケートされるオブジェクトの現在の状態を判断するのに役立ちます。レプリケート元オブジェクトのレプリケーションステータスは、`PENDING`、`COMPLETED`または`FAILED`のいずれかを返します。レプリカのレプリケーションステータスが`REPLICA`に返されます。

レプリケーションステータスの概要

レプリケーションシナリオでは、レプリケーションを設定するレプリケート元バケットと、S3 on Outposts がオブジェクトをレプリケートするレプリケート先バケットを使用します。これらのバケットからオブジェクト (`GetObject` を使用) またはオブジェクトメタデータ (`HeadObject` を使用) をリクエストすると、S3 on Outposts はレスポンスとして以下のように `x-amz-replication-status` ヘッダーを返します。

- レプリケート元バケットのオブジェクトをリクエストする場合、リクエストしたオブジェクトがレプリケーション対象であると、S3 on Outposts は `x-amz-replication-status` ヘッダーを返します。

例えば、レプリケーション設定でオブジェクトプレフィックス TaxDocs を指定して、キー名のプレフィックス TaxDocs が付いたオブジェクトのみをレプリケートするように S3 on Outposts に指示しているとします。このキー名のプレフィックスを持つ、アップロードしたすべてのオブジェクト (TaxDocs/document1.pdf など) がレプリケートされます。このキー名のプレフィックスが付いたオブジェクトのリクエストでは、S3 on Outposts が、オブジェクトのレプリケーション状態が PENDING、COMPLETED、または FAILED の値のいずれかの x-amz-replication-status ヘッダーを返します。

> **Note**
>
> オブジェクトをアップロードした後で、オブジェクトのレプリケーションに失敗した場合、レプリケーションを再試行できません。もう一度オブジェクトをアップロードする必要があります。レプリケーションロールの許可またはバケットの許可がないなどの問題がある場合、オブジェクトは FAILED の状態に移行します。バケットや Outpost が使用できないなどの一時的な障害が発生した場合、レプリケーションのステータスは FAILED にはならず、PENDING のままになります。リソースがオンラインに戻ると、S3 on Outposts はこれらのオブジェクトのレプリケーションを再開します。

- レプリケート先バケットからオブジェクトをリクエストした場合、リクエストされたオブジェクトが S3 on Outposts によって作成されたレプリカであるときに、S3 on Outposts は値が REPLICA である x-amz-replication-status ヘッダーを返します。

> **Note**
>
> レプリケーションが有効になっているレプリケート元バケットからオブジェクトを削除する前に、削除する前にオブジェクトのレプリケーションステータスをチェックして、そのオブジェクトがレプリケートされていることを確認します。

Outposts の Amazon S3 レプリカ変更の同期が有効になっている場合のレプリケーションステータス

レプリケーションルールが S3 on Outposts レプリカの変更を有効にすると、レプリカは REPLICA 以外のステータスをレポートできます。メタデータの変更がレプリケート中の場合は、レプリカの x-amz-replication-status ヘッダーは PENDING を返します。レプリカ変更の同期がメタデータのレプリケートに失敗した場合、レプリカのヘッダーは FAILED を返します。メタデータが正しくレプリケートされると、レプリカのヘッダーは値 REPLICA を返します。

レプリケーションのトラブルシューティング

レプリケーションを設定した後にオブジェクトレプリカがレプリケート先 Amazon S3 on Outposts バケットに表示されない場合は、これらのトラブルシューティングのヒントを使用して問題を特定し、修正してください。

- S3 on Outposts がオブジェクトをレプリケートするために要する時間はさまざまな要因に依存します (レプリケート元とレプリケート先の Outposts、オブジェクトのサイズなど)。

  ソースオブジェクトのレプリケーションステータスを確認することもできます。オブジェクトのレプリケーションステータスが `PENDING` の場合は、S3 on Outposts がレプリケーションを完了していません。オブジェクトのレプリケーションステータスが `FAILED` の場合は、レプリケート元バケットのレプリケーション設定を確認してください。
- レプリケート元バケットのレプリケーション設定について、以下を確認します。
  - レプリケート先バケットのアクセスポイント Amazon リソースネーム (ARN) が正しい。
  - キー名のプレフィックスが正しい。たとえば、`Tax` というプレフィックスが付いたオブジェクトをレプリケートする設定にした場合、`Tax/document1` や `Tax/document2` のようなキー名のオブジェクトのみがレプリケートされます。キー名が "`document3`" のオブジェクトはレプリケートされません。
  - ステータスは`Enabled`です。
- どのバケットでもバージョニングが一時停止されていないことを確認します。レプリケート元とレプリケート先の両方のバケットで、バージョニングを有効にする必要があります。
- レプリケート先バケットが他の AWS アカウントによって所有されている場合、バケット所有者が、レプリケート元バケットの所有者に対してオブジェクトのレプリケーションを許可するバケットポリシーをレプリケート先バケットに設定していることを確認します。例については、「レプリケーション元とレプリケーション先の Outposts バケットが異なる AWS アカウントによって所有されている場合のアクセス許可の付与」を参照してください。
- オブジェクトのレプリカがレプリケート先バケットに表示されない場合、以下がレプリケーションを妨げている可能性があります。
  - レプリケート元バケットのオブジェクト自体が別のレプリケーション設定によって作成されたレプリカである場合、S3 on Outposts はそのオブジェクトをレプリケートしません。例えば、バケット A からバケット B へ、バケット B からバケット C へのレプリケーション設定を設定した場合、S3 on Outposts はバケット B にあるオブジェクトレプリカをバケット C へレプリケートしません。

バケット A のオブジェクトをバケット B とバケット C にレプリケートする場合は、ソースバケットのレプリケーション設定の異なるレプリケーションルールで複数のバケット宛先を設定します。例えば、ソースバケット A に 2 つのレプリケーションルールを作成します。1 つのルールは宛先バケット B にレプリケートし、もう 1 つのルールは宛先バケット C にレプリケートします。

- レプリケート元バケット所有者は、オブジェクトをアップロードするための許可を他の AWS アカウントに付与できます。デフォルトでは、レプリケート元バケット所有者は、他のアカウントによって作成されたオブジェクトに対するアクセス許可を持ちません。レプリケーション設定では、レプリケート元バケット所有者がアクセス許可を持つオブジェクトのみがレプリケートされます。レプリケーションの問題を回避するために、レプリケート元バケット所有者は、条件付きでオブジェクトを作成するための他の AWS アカウント アクセス許可を付与し、それらのオブジェクトに対する明示的なアクセス許可を要求することができます。ポリシーの例については、バケット所有者はフルコントロール権限を持ちながら、オブジェクトをアップロードするためのクロスアカウントアクセス許可を付与するを参照してください。

- レプリケーション設定に、特定のタグを持つオブジェクトのサブセットをレプリケートするためのルールを追加するとします。この場合、S3 on Outposts がオブジェクトをレプリケートするには、オブジェクトの作成時に特定のタグキーと値を割り当てる必要があります。まずオブジェクトを作成して、それから既存のオブジェクトにタグを追加した場合、S3 on Outposts はそのオブジェクトをレプリケートしません。
- バケットポリシーで次のいずれかのアクションでレプリケーションのロールへのアクセスを拒否した場合、レプリケーションは失敗します。

  レプリケート元のバケット

  ```
  "s3-outposts:GetObjectVersionForReplication",
  "s3-outposts:GetObjectVersionTagging"
  ```

  レプリケート先バケット:

  ```
  "s3-outposts:ReplicateObject",
  "s3-outposts:ReplicateDelete",
  "s3-outposts:ReplicateTags"
  ```

- EventBridge は、オブジェクトがレプリケート先の Outposts にレプリケートされない場合に通知できます。詳細については、「S3 Replication on Outposts に EventBridge を使用する」を参照してください。

S3 Replication on Outposts に EventBridge を使用する

Amazon S3 on Outposts は、Amazon EventBridge と統合されており、`s3-outposts` ネームスペースを使用します。EventBridge は、アプリケーションをさまざまなソースからのデータに接続するために使用できるサーバーレスのイベントバスサービスです。詳細については、「Amazon EventBridge ユーザーガイド」の「Amazon EventBridge とは」を参照してください。

レプリケーション設定の問題のトラブルシューティングを支援するために、Amazon EventBridge を設定して、レプリケーションの失敗イベントに関する通知を受け取ることもできます。EventBridge は、オブジェクトがデスティネーション Outposts にレプリケートされない場合に通知できます。レプリケートされるオブジェクトの現在の状態の詳細については、「レプリケーションステータスの概要」を参照してください。

Outposts バケット内で特定のイベントが発生するたびに S3 on Outposts は EventBridge にイベントを送信できます。他の宛先とは異なり、配信するイベントタイプを選択する必要はありません。また、EventBridge ルールを使用すると、イベントを追加のターゲットにルートできます。EventBridge を有効にすると、S3 on Outposts は次のすべてのイベントを EventBridge に送信します。

イベントタイプ	説明	名前空間
`Operation FailedReplication`	レプリケーションルール内のオブジェクトのレプリケーションが失敗しました。S3 Replication on Outposts の詳細については、「EventBridge を使用して S3 Replication on Outposts の障害理由を表示する」を参照してください。	`s3-outposts`

EventBridge を使用して S3 Replication on Outposts の障害理由を表示する

次の表は、Outposts の S3 レプリケーションが失敗した理由を示しています。EventBridge ルールを設定して、Amazon Simple Queue Service (Amazon SQS)、Amazon Simple Notification Service (Amazon SNS)、AWS Lambda、または Amazon CloudWatch Logs を介して失敗の理由を公開して表示できます。EventBridge でこれらのリソースを使用するために必要なアクセス許可の詳細については、「EventBridge のリソースベースのポリシーを使用する」を参照してください。

レプリケーションの失敗の理由	説明
AssumeRoleNotPermitted	S3 on Outposts は、レプリケーション設定で指定されている AWS Identity and Access Management (IAM) ロールを引き継ぐことはできません。
DstBucketNotFound	S3 on Outposts は、レプリケーション設定で指定されたレプリケート先バケットを見つけることができません。
DstBucketUnversioned	Outposts レプリケート先バケットでバージョニングが有効になっていません。S3 Replication on Outposts でオブジェクトをレプリケートするには、レプリケート先バケットのバージョニングを有効にします。
DstDelObjNotPermitted	S3 on Outposts は、レプリケート先バケットに削除をレプリケートできません。レプリケート先バケットに対する s3-outposts:ReplicateDelete 権限がない可能性があります。
DstMultipartCompleteNotPermitted	S3 on Outposts は、レプリケート先バケットでオブジェクトのマルチパートアップロードを完了できません。レプリケート先バケットに対する s3-outposts:ReplicateObject 権限がない可能性があります。
DstMultipartInitNotPermitted	S3 on Outposts は、レプリケート先バケットへのオブジェクトのマルチパートアップロードを開始できません。レプリケート先バケットに対する s3-outposts:ReplicateObject 権限がない可能性があります。

レプリケーションの失敗の理由	説明
DstMultipartPartUploadNotPermitted	S3 on Outposts は、レプリケート先バケットにマルチパートオブジェクトをアップロードできません。レプリケート先バケットに対する s3-outposts:ReplicateObject 権限がない可能性があります。
DstOutOfCapacity	Outpost に S3 ストレージ容量が不足しているため、S3 on Outposts は、デスティネーションアウトポストにレプリケートできません。
DstPutObjNotPermitted	S3 on Outposts は、レプリケート先バケットにオブジェクトをレプリケートできません。レプリケート先バケットに対する s3-outposts:ReplicateObject 権限がない可能性があります。
DstPutTaggingNotPermitted	S3 on Outposts は、レプリケート先バケットにオブジェクトタグをレプリケートできません。レプリケート先バケットに対する s3-outposts:ReplicateObject 権限がない可能性があります。
DstVersionNotFound	S3 on Outposts は、そのオブジェクトバージョンのメタデータをレプリケートするためにレプリケート先バケットで必要なオブジェクトバージョンを見つけることができません。
SrcBucketReplicationConfigMissing	S3 on Outposts は、ソースの Outposts バケットに関連付けられているアクセスポイントのレプリケーション設定を見つけることができません。

レプリケーションの失敗の理由	説明
SrcGetObjNotPermitted	S3 on Outposts は、レプリケーションのレプリケート元バケットにあるオブジェクトにアクセスできません。レプリケート元バケットに対する s3-outposts:GetObjectVersionForReplication 権限がない可能性があります。
SrcGetTaggingNotPermitted	S3 on Outposts は、レプリケート元バケットからオブジェクトタグ情報にアクセスできません。レプリケート元バケットに対する s3-outposts:GetObjectVersionTagging 権限がない可能性があります。
SrcHeadObjectNotPermitted	S3 on Outposts は、レプリケート元バケットからオブジェクトメタデータを取得できません。レプリケート元バケットに対する s3-outposts:GetObjectVersionForReplication 権限がない可能性があります。
SrcObjectNotEligible	このオブジェクトはレプリケーションの対象外です。オブジェクトまたはそのオブジェクトタグがレプリケーション設定と一致しません。

レプリケーションのトラブルシューティングの詳細については、以下のトピックを参照してください。

- IAM ロールの作成
- レプリケーションのトラブルシューティング

CloudWatch での EventBridge のモニタリング

モニタリングには、Amazon EventBridge が Amazon CloudWatch と統合します。EventBridge は自動的に CloudWatch に毎分メトリクスを送信します。これらのメトリクスには、ルールに一致

したイベントの数と、ルールによってターゲットが呼び出された回数が含まれます。Eventbridge でルールが実行されると、このルールに関連付けられているすべてのターゲットが呼び出されます。EventBridge の動作は、CloudWatch を通じて次の方法でモニタリングできます。

- EventBridge ルールで利用可能な EventBridge メトリクスは、CloudWatch ダッシュボードからモニタリングできます。次に、CloudWatch アラームなどの CloudWatch 機能を使用して、特定のメトリクスにアラームを設定できます。これらのメトリクスがアラームで指定したカスタムしきい値に達すると、通知が届き、それに応じてアクションを実行できます。
- Amazon CloudWatch Logs を EventBridge ルールのターゲットとして設定できます。次に、EventBridge がログストリームを作成し、CloudWatch Logs がログエントリとしてイベントからテキストを保存します。詳細については、「EventBridge の CloudWatch Logs のアクセス許可」を参照してください。

EventBridge イベント配信のデバッグとイベントのアーカイブの詳細については、次のトピックを参照してください。

- イベントの再試行ポリシーとデッドレターキューの使用
- EventBridge イベントのアーカイブ

AWS RAM を使用して S3 on Outposts を共有する

Simple Storage Service (Amazon S3) on Outposts は、AWS Resource Access Manager (AWS RAM) を使用して、組織内の複数のアカウント間で S3 容量の共有をサポートしています。。S3 on Outposts 共有を使用すると、他のユーザーが Outpost でバケット、エンドポイント、およびアクセスポイントを作成および管理できるようになります。

このトピックでは、AWS RAM を使用して S3 on Outposts と関連するリソースを、お客様の AWS 組織の別の AWS アカウントと共有する方法をご紹介します。

前提条件

- Outpost 所有者アカウントには AWS Organizations で設定された組織が存在します。詳細については、「AWS Organizations ユーザーガイド」で「組織を作成する」を参照してください。
- 組織には S3 on Outposts の容量を共有する AWS アカウント が含まれます。詳細については、「AWS Organizations ユーザーガイド」の「AWS アカウント への招待の送信」を参照してください。

- 共有する以下のオプションのいずれかを選択します。エンドポイントにもアクセスできるように、2 番目のリソース (サブネットまたは Outposts) を選択する必要があります。エンドポイントは、S3 on Outposts に格納されたデータにアクセスするためのネットワーク要件です。

オプション 1	オプション 2
S3 on Outposts ユーザーが Outposts とアクセスポイントにバケットを作成し、それらのバケットにオブジェクトを追加できるようにします。	S3 on Outposts ユーザーが Outposts とアクセスポイントにバケットを作成し、それらのバケットにオブジェクトを追加できるようにします。
サブネット ユーザーが仮想プライベートクラウド (VPC) とサブネットに関連付けられているエンドポイントを使用できるようにします。	Outposts ユーザーが S3 容量チャートと AWS Outposts コンソールのホームページを参照できるようにします。また、共有の Outposts にサブネットを作成し、エンドポイントを作成することもできます。

手順

1. Outpost を所有する AWS アカウント を使用して AWS Management Console にサインインし、https://console.aws.amazon.com/ram で AWS RAM コンソールを開きます。
2. AWS RAM で AWS Organizations との共有が有効になっていることを確認します。詳細については、「AWS RAM ユーザーガイド」の「AWS Organizations 内のリソース共有を有効化する」を参照してください。
3. 前提条件で、オプション 1 またはオプション 2 を使用してリソース共有を作成します。複数の S3 on Outposts リソースがある場合は、共有するリソースの Amazon リソースネーム (ARN) を選択します。エンドポイントを有効にするには、サブネットまたは Outpost を共有します。

   リソース共有の作成の詳細については、「AWS RAM ユーザーガイド」の「リソース共有を作成する」を参照してください。
4. リソースを共有した AWS アカウント は、S3 on Outposts を使用できるようになります。前提条件で選択したオプションに応じて、アカウントユーザーに次の情報を提供します。

オプション 1	オプション 2
Outpost ID	Outpost ID
VPC ID。	
サブネット ID	
セキュリティグループの ID	

> **Note**
>
> ユーザーは、AWS RAM コンソール、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して、リソースが共有されていることを確認できます。ユーザーは、get-resource-shares CLI コマンドを使用して、既存のリソース共有を表示できます。

使用例

S3 on Outposts リソースを別のアカウントと共有した後、そのアカウントは Outpost のバケットとオブジェクトを管理できます。[Subnets] (サブネット) リソース共有した場合、そのアカウントは、作成したエンドポイントを使用できます。次の例では、ユーザーが AWS CLI を使用し、これらのリソースを共有した後、Outpost とやり取るする方法についてご紹介します。

Example : バケットの作成

次の例では、Outpost *op-01ac5d28a6a232904* に *example-s3-bucket1* という名前のバケットを作成します。このコマンドを使用する前に、それぞれの *user input placeholder* をユースケースに適した値に置き換えてください。

```
aws s3control create-bucket --bucket example-s3-bucket1 --outpost-
id op-01ac5d28a6a232904
```

このコマンドの詳細については、「AWS CLI リファレンス」の「create-bucket」を参照してください。

Example : アクセスポイントの作成

次の例では、次の表に示すパラメータ例を使用して、Outpost 上にアクセスポイントを作成します。このコマンドを使用する前に、これらの *user input placeholder* 値と AWS リージョンコードをユースケースに適した値に置き換えてください。

[Parameter] (パラメータ)	値
アカウント ID	*111122223333*
アクセスポイント名	*example-outpost-access-point*
Outpost ID	*op-01ac5d28a6a232904*
Outpost バケット名	*example-s3-bucket1*
VPC ID	*vpc-1a2b3c4d5e6f7g8h9*

> **Note**
>
> アカウント ID パラメータは、AWS アカウントバケット所有者 (共有ユーザー) の ID である必要があります。

```
aws s3control create-access-point --account-id 111122223333 --name example-outpost-
access-point \
--bucket arn:aws:s3-outposts:us-east-1:111122223333:outpost/op-01ac5d28a6a232904/
bucket/example-s3-bucket1 \
--vpc-configuration VpcId=vpc-1a2b3c4d5e6f7g8h9
```

このコマンドの使用に関する詳細については、「AWS CLI リファレンス」の「create-access-point」を参照してください。

Example : オブジェクトのアップロード

次の例では、ユーザーのローカルファイルシステムから、AWS アカウント *111122223333* が所有しているアクセスポイント *example-outpost-access-point* on the Outpost*op-01ac5d28a6a232904* を経由し、*images/my_image.jpg* という名前のオブジェク

トへ、ファイル *my_image.jpg* をアップロードします。このコマンドを使用する前に、これらの *user input placeholder* 値と AWS リージョンコードをユースケースに適した値に置き換えてください。

```
aws s3api put-object --bucket arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/example-outpost-access-
point \
--body my_image.jpg --key images/my_image.jpg
```

このコマンドの詳細については、「AWS CLI リファレンス」の「put-object」を参照してください。

> **Note**
>
> このオペレーションの結果、[Resource not found] (リソースが見つからない) エラーまたは応答がない場合、VPC に共有エンドポイントがない可能性があります。
> 共有エンドポイントがあるかどうかを確認するには、list-shared-endpoints AWS CLI コマンドを使用します。共有エンドポイントがない場合は、Outpost の所有者と連携して作成します。詳細については、「Amazon Simple Storage Service API リファレンス」の「ListSharedEndpoints」を参照してください。

Example : エンドポイントの作成

次の例では、共有の Outposts にエンドポイントを作成します。このコマンドを使用する前に、Outpost ID、サブネット ID、およびセキュリティグループ ID の *user input placeholder* 値をユースケースに適した値に置き換えます。

> **Note**
>
> ユーザーがこの操作を実行できるのは、リソース共有に Outposts リソースが含まれている場合に限ります。

```
aws s3outposts create-endpoint --outposts-id op-01ac5d28a6a232904 --subnet-id XXXXXX --
security-group-id XXXXXXX
```

このコマンドの詳細については、「AWS CLI リファレンス」の「create-endpoint」を参照してください。

S3 on Outposts を使用するその他の AWS のサービス

AWS Outposts に対してローカルで実行するその他の AWS のサービス も、Amazon S3 on Outposts の容量を使用できます。Amazon CloudWatch では、`S3Outposts` 名前空間には、S3 on Outposts 内のバケットの詳細なメトリクスが表示されますが、これらのメトリクスには、他 AWS のサービスの使用は含まれません。他の AWS のサービス によって使用される S3 on Outposts の容量を管理するには、次の表の情報を参照してください。

AWS のサービス	説明	詳細情報
Amazon S3	すべての直接的な S3 on Outposts の使用量には、一致するアカウントおよびバケットの CloudWatch メトリクスがあります。	メトリクスを参照
Amazon Elastic Block Store (Amazon EBS)	Outposts 上の Amazon EBS の場合、スナップショットのデスティネーションとして AWS Outpost を選択して、S3 on Outpost にローカルに保存できます。	詳細はこちら
Amazon Relational Database Service (Amazon RDS)	Amazon RDS ローカルバックアップを使用して、RDS バックアップを Outpost にローカルに保存できます。	詳細はこちら

S3 on Outposts のモニタリング

Amazon S3 on Outposts を使用すると、AWS Outposts で S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを簡単に保存および取得できます。S3 on Outposts は、新しいストレージクラス、S3 Outposts (OUTPOSTS) を提供し、これは Amazon S3 API を使用し、AWS Outposts 上の複数のデバイスやサーバー間でデータを永続的かつ冗長的に保存するように設計されています。仮想プライベートクラウド (VPC) を介したアクセスポイントとエンドポイント接続を使用して、Outposts バケットと通信します。Outposts バケットでは、Amazon S3 と同じ API と機能 (アクセスポリシー、暗号化、タグ付けなど) を使用できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。詳細については、「Amazon S3 on Outposts とは」を参照してください。

Amazon S3 on Outposts ストレージ容量のモニタリングの詳細については、以下のトピックを参照してください。

トピック

- Amazon CloudWatch メトリクスを使用した S3 on Outposts 容量の管理
- Amazon CloudWatch Events を使用した S3 on Outposts イベント通知の受信
- AWS CloudTrail ログで S3 on Outposts をモニタリングする

Amazon CloudWatch メトリクスを使用した S3 on Outposts 容量の管理

S3 の容量を管理するために、ストレージ使用率が特定のしきい値を超えたときにその旨を知らせる CloudWatch アラートを作成するようお勧めします。S3 on Outposts の CloudWatch メトリクスの詳細については、「CloudWatch メトリクス」を参照してください。Outpost にオブジェクトを保存するための十分なスペースがない場合、API は容量不足の例外 (ICE) を返します。容量を空けるには、明示的なデータ削除をトリガーする CloudWatch アラームを作成するか、ライフサイクルの有効期限ポリシーを使用してオブジェクトを期限切れにすることができます。削除前にデータを保存するには、AWS DataSync を使用して Amazon S3 on Outposts のバケットから AWS リージョン にある S3 バケットにデータをコピーできます。DataSync の使用の詳細については、「AWS DataSync ユーザーガイド」の「AWS DataSync の開始方法」を参照してください。

CloudWatch メトリクス

`S3Outposts` 名前空間には、Amazon S3 on Outposts バケットのための以下のメトリクスが含まれます。プロビジョニングされた S3 on Outposts バイトの合計数、オブジェクトで使用可能な合計空きバイト数、および特定のバケットのすべてのオブジェクトの合計サイズをモニタリングできます。バケットまたはアカウント関連のメトリクスは、S3 を直接使用する場合すべてに存在しています。Amazon Elastic Block Store のローカルスナップショットや Amazon Relational Database Service バックアップを Outpost に保存するなど、S3 を間接的に使用することで S3 の容量が消費されますが、バケットやアカウント関連のメトリクスには含まれません。Amazon EBS スナップショットの詳細については、「Amazon EBS Local Snapshots on Outposts」(Outposts の Amazon EBS ローカルスナップショット) を参照してください。Amazon EBS コストレポートを確認するには、https://console.aws.amazon.com/billing/ をご覧ください。

> **Note**
>
> S3 on Outposts は、次のメトリクスのみをサポートし、他の Amazon S3 メトリクスはサポートしません。

S3 on Outposts の容量は限られているため、ストレージ使用率が特定のしきい値を超えたときに警告する CloudWatch アラートを作成するようお勧めします。

メトリクス	説明	期間	単位	タイプ
OutpostT talBytes	Outpost のプロビジョニングされた合計キャパシティー (バイト単位)。	5 分	バイト	S3 on Outposts
OutpostF eeBytes	お客様のデータを保存するために Outpost で使用可能な空きバイト数。	5 分	バイト	S3 on Outposts
BucketUs dBytes	指定されたバケットのすべてのオブジェクトの合計サイズ。	5 分	バイト	S3 on Outposts。S3 での直接利用のみ。
AccountU edBytes	指定された Outposts アカウントのすべてのオブジェクトの合計サイズ。	5 分	バイト	S3 on Outposts。S3 での直接利用のみ。
BytesPen ingRepli ation	特定のレプリケーションルールについて、レプリケーションが保留中のオブジェクトの合計バイト数。レプリケーションメトリクスを有効にする方法の詳細については、「Outposts 間のレプリケーションルールの作成」を参照してください。	5 分	バイト	オプション。Outposts の S3 レプリケーションの場合。
Operatio sPendin eplicati n	特定のレプリケーションルールについて、レプリケーションが保留中のオペレーションの総数。レプリケーションメトリクスを有効にする方法の	5 分	カウント	オプション。Outposts の S3 レプリケーションの場合。

メトリクス	説明	期間	単位	タイプ
	詳細については、「Outposts 間のレプリケーションルールの作成」を参照してください。			
ReplicationLatency	特定のレプリケーションルールで、レプリケーション先バケットがレプリケート元バケットよりも遅れている現在の遅延秒数。レプリケーションメトリクスを有効にする方法の詳細については、「Outposts 間のレプリケーションルールの作成」を参照してください。	5 分	[秒]	オプション。Outposts の S3 レプリケーションの場合。

Amazon CloudWatch Events を使用した S3 on Outposts イベント通知の受信

CloudWatch Events を使用して、Amazon S3 on Outposts API イベントのルールを作成できます。ルールを作成するときに、Amazon Simple Queue Service (Amazon SQS)、Amazon Simple Notification Service (Amazon SNS)、および AWS Lambda を含む、サポートされているすべての CloudWatch ターゲットを通じて通知を受け取ることを選択できます。詳細については、「Amazon CloudWatch Events ユーザーガイド」のCloudWatch Events のターゲットとなるAWS サービスのリスト」を参照してください。S3 on Outposts で操作するターゲットサービスを選択するには、「Amazon CloudWatch Events ユーザーガイド」の「AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成」を参照してください。

> **Note**
>
> S3 on Outposts オブジェクトオペレーションでは、CloudTrail によって送信される AWS API コールイベントは、それらのイベントを受信するように証跡 (オプションでイベントセ

レクターが付いています) を設定している場合にのみ、ルールに一致します。詳細については、AWS CloudTrail ユーザーガイドのCloudTrail ログファイルの操作を参照してください。

Example

以下に、DeleteObject オペレーションのサンプルルールを示します。このサンプルルールを使用するには、*example-s3-bucket1* を S3 on Outposts バケットの名前と置き換えます。

```
{
  "source": [
    "aws.s3-outposts"
  ],
  "detail-type": [
    "AWS API call through CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "s3-outposts.amazonaws.com"
    ],
    "eventName": [
      "DeleteObject"
    ],
    "requestParameters": {
      "bucketName": [
        "example-s3-bucket1"
      ]
    }
  }
}
```

AWS CloudTrail ログで S3 on Outposts をモニタリングする

Amazon S3 on Outposts は AWS CloudTrail と統合されています。このサービスは、ユーザーやロール、または S3 on Outposts の AWS のサービス によって実行されたアクションを記録するサービスです。AWS CloudTrail を使用して、S3 on Outposts のバケットレベルおよびオブジェクトレベルのリクエストに関する情報を取得し、S3 on Outposts イベントアクティビティを監査してログに記録できます。すべての Outposts バケットまたは特定の Outposts バケットのリストの CloudTrail データイベントを有効にするには、CloudTrail で手動によって証跡を作成する必要があります。CloudTrail ログファイルエントリの詳細については、「S3 on Outposts ログファイルエントリ」を参照してください。

Note

- ベストプラクティスとして、AWS CloudTrail データイベント Outposts バケットに対してライフサイクルポリシーを作成することをお勧めします。ログファイルを監査する必要がある期間が経過したらログファイルを定期的に削除するように、ライフサイクルポリシーを設定します。これにより、各クエリで Amazon Athena が分析するデータの量が減ります。詳細については、「バケットにライフサイクル設定を設定する」を参照してください。
- CloudTrail ログをクエリする方法の例については、AWS ビッグデータブログの記事 Analyze Security, Compliance, and Operational Activity Using AWS CloudTrail and Amazon Athena を参照してください。

S3 on Outposts バケットでオブジェクトの CloudTrail ログ記録を有効にするには

Amazon S3 コンソールを使用し、AWS CloudTrail 証跡を設定して、Amazon S3 on Outposts バケット内のオブジェクトのデータイベントをログに記録できます。CloudTrail では、`GetObject`、`DeleteObject`、`PutObject` など、S3 on Outposts オブジェクトレベルの API オペレーションのログ記録がサポートされます。これらのイベントは、データイベントと呼ばれます。

デフォルトでは、CloudTrail 証跡はデータイベントを記録しません。ただし、指定した S3 on Outposts バケットのデータイベントを記録するか、AWS アカウント の S3 on Outposts バケット上のすべてのデータイベントを記録するように証跡を設定できます。詳細については、「AWS CloudTrail を使用した Amazon S3 API コールのログ記録」を参照してください。

CloudTrail では、CloudTrail イベント履歴にデータイベントが設定されません。さらに、すべての S3 on Outposts バケットレベルの API オペレーションが CloudTrail イベント履歴に入力されるわけではありません。CloudTrail ログのクエリ方法の詳細については、AWS ナレッジセンターの「Amazon CloudWatch Logs のフィルターパターンと Amazon Athena を使用して CloudTrail ログをクエリする」を参照してください。

S3 on Outposts バケットのデータイベントをログに記録するように証跡を設定する場合、AWS CloudTrail コンソールまたは Amazon S3 コンソールのいずれかを使用できます。AWS アカウント 内のすべての S3 on Outposts バケットのデータイベントを記録するように証跡を設定する場合は、CloudTrail コンソールを使用する方が簡単です。CloudTrail コンソールを使用して S3 on Outposts データイベントを記録するように証跡を設定する方法については、AWS CloudTrail ユーザーガイドの「データイベント」を参照してください。

> **Important**
>
> 追加の変更がイベントデータに適用されます。詳細については、「AWS CloudTrail 料金表」を参照してください。

以下の手順では、Amazon S3 コンソールを使用して、CloudTrail の証跡で S3 on Outposts バケットのデータイベントの記録を設定する方法を示します。

> **Note**
>
> バケットを作成する AWS アカウントが、そのバケットを所有し、AWS CloudTrail に送信する S3 on Outposts データイベントを設定できる唯一のアカウントです。

S3 on Outposts バケットでオブジェクトの CloudTrail データイベントの記録を有効にするには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Outposts buckets] (Outposts バケット) を選択します。
3. CloudTrail を使用してログを記録するデータイベントがある Outposts バケットの名前を選択します。
4. [プロパティ] を選択します。
5. [AWS CloudTrail データイベント] セクションに進み、[CloudTrail で設定] を選択します。

   AWS CloudTrail コンソールが開きます。

   新しい CloudTrail の証跡を作成するか、既存の証跡を再利用して、証跡に記録するように S3 on Outposts データイベントを設定できます。
6. CloudTrail コンソールの[ダッシュボード] ページで、[証跡の作成] を選択します。
7. [ステップ 1 証跡の属性の選択] ページで、証跡の名前を入力し、証跡ログを保存する S3 バケットを選択し、その他の必要な設定を指定して、[次へ] を選択します。
8. [ステップ 2 の ログイベントの選択] ページの [イベントタイプ] で、[データイベント] を選択します。

   [データイベントタイプ] では、[S3 Outposts] を選択します。[Next] を選択します。

> **Note**
>
> - トレイルを作成して S3 on Outposts のデータイベントのログ記録を設定する場合、データイベントタイプを正しく指定する必要があります。
>   - CloudTrail コンソールを使用する場合は、[データイベントタイプ] で [S3 Outposts] を選択します。CloudTrail コンソールで証跡を作成する方法については、AWS CloudTrail ユーザーガイドのコンソールで証跡を作成および更新するを参照してください。CloudTrail コンソールで S3 on Outposts データイベントのログ記録を設定する方法については、AWS CloudTrail ユーザーガイドの 「Amazon S3 オブジェクトのデータイベントをログに記録する」を参照してください。
>   - AWS Command Line Interface (AWS CLI) または AWS SDK を使用する場合は、`resources.type` フィールドを `AWS::S3Outposts::Object` に設定します。AWS CLI を使用して S3 on Outposts データイベントをログに記録する方法の詳細については、AWS CloudTrail ユーザーガイドの「S3 on Outposts イベントのログ記録」を参照してください。
> - S3 on Outposts バケットのデータイベントをログに記録するための証跡の設定に CloudTrail コンソールまたは Amazon S3 コンソールを使用する場合、そのバケットに対してオブジェクトレベルのロギングが有効化されていることが Amazon S3 コンソールに表示されます。

9. [ステップ 3 の確認と作成] ページで、設定した証跡属性とログイベントを確認します。次に、[証跡の作成]を選択します。

S3 on Outposts バケットでオブジェクトの CloudTrail データイベントの記録を無効にするには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudtrail/ で CloudTrail コンソールを開きます。
2. 左のナビゲーションペインで、[証跡] を選択します。
3. S3 on Outposts バケットのイベントを記録するために作成した証跡の名前を選択します。
4. 証跡の詳細ページで、右上隅にある [ログインを停止] を選択します。
5. 表示されたダイアログボックスで [ログインを停止] を選択します。

Amazon S3 on Outposts での開発

Amazon S3 on Outposts を使用すると、AWS Outposts で S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを簡単に保存および取得できます。S3 on Outposts は、新しいストレージクラス、S3 Outposts (OUTPOSTS) を提供し、これは Amazon S3 API を使用し、AWS Outposts 上の複数のデバイスやサーバー間でデータを永続的かつ冗長的に保存するように設計されています。仮想プライベートクラウド (VPC) を介したアクセスポイントとエンドポイント接続を使用して、Outposts バケットと通信します。Outposts バケットでは、Amazon S3 と同じ API と機能 (アクセスポリシー、暗号化、タグ付けなど) を使用できます。AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。詳細については、「Amazon S3 on Outposts とは」を参照してください。

以下のトピックでは、S3 on Outposts での開発について説明します。

トピック

- Amazon S3 on Outposts の API オペレーション
- SDK for Java を使用して、S3 on Outposts の S3 コントロールクライアントを設定する
- IPv6 経由で S3 on Outposts へのリクエストを行う

Amazon S3 on Outposts の API オペレーション

このトピックでは、Amazon S3 Outposts で使用できる Amazon S3、Amazon S3 コントロール、および Amazon S3 on Outposts API オペレーションの一覧を示します。

トピック

- オブジェクト管理のための Amazon S3 API オペレーション
- バケット管理のための Amazon S3 コントロール API オペレーション
- Outposts を管理するための S3 on Outposts API オペレーション

オブジェクト管理のための Amazon S3 API オペレーション

S3 on Outposts は、Amazon S3 と同じオブジェクト API オペレーションを使用するように設計されています。Outpost バケット内の任意のオブジェクトにアクセスするには、アクセスポイントを使用する必要があります。S3 on Outposts でオブジェクト API オペレーションを使用するときは、Outposts アクセスポイントの Amazon リソースネーム (ARN) またはアクセスポイントエイリア

スのいずれかを指定します。アクセスポイントエイリアスの詳細については、「S3 on Outposts アクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

Amazon S3 on Outposts では、以下の Amazon S3 API のオペレーションがサポートされています。

- AbortMultipartUpload
- CompleteMultipartUpload
- CopyObject
- CreateMultipartUpload
- DeleteObject
- DeleteObjects
- DeleteObjectTagging
- GetObject
- GetObjectTagging
- HeadBucket
- HeadObject
- ListMultipartUploads
- ListObjects
- ListObjectsV2
- ListObjectVersions
- ListParts
- PutObject
- PutObjectTagging
- UploadPart
- UploadPartCopy

バケット管理のための Amazon S3 コントロール API オペレーション

S3 on Outposts では、バケット操作のために以下の Amazon S3 コントロール API オペレーションがサポートされています。

- CreateAccessPoint
- CreateBucket

- DeleteAccessPoint
- DeleteAccessPointPolicy
- DeleteBucket
- DeleteBucketLifecycleConfiguration
- DeleteBucketPolicy
- DeleteBucketReplication
- DeleteBucketTagging
- GetAccessPoint
- GetAccessPointPolicy
- GetBucket
- GetBucketLifecycleConfiguration
- GetBucketPolicy
- GetBucketReplication
- GetBucketTagging
- GetBucketVersioning
- ListAccessPoints
- ListRegionalBuckets
- PutAccessPointPolicy
- PutBucketLifecycleConfiguration
- PutBucketPolicy
- PutBucketReplication
- PutBucketTagging
- PutBucketVersioning

Outposts を管理するための S3 on Outposts API オペレーション

S3 on Outposts では、エンドポイントを管理するために、以下の Amazon S3 on Outposts API オペレーションがサポートされています。

- CreateEndpoint
- DeleteEndpoint
- ListEndpoints

- ListOutpostsWithS3
- ListSharedEndpoints

SDK for Java を使用して、S3 on Outposts の S3 コントロールクライアントを設定する

次の例では、AWS SDK for Java を使用して、Amazon S3 on Outposts の Amazon S3 コントロールクライアントを設定します。この例を実行するには、それぞれの *user input placeholder* をユーザー自身の情報に置き換えます。

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;

public AWSS3Control createS3ControlClient() {

    String accessKey = AWSAccessKey;
    String secretKey = SecretAccessKey;
    BasicAWSCredentials awsCreds = new BasicAWSCredentials(accessKey, secretKey);

    return AWSS3ControlClient.builder().enableUseArnRegion()
            .withCredentials(new AWSStaticCredentialsProvider(awsCreds))
            .build();

}
```

IPv6 経由で S3 on Outposts へのリクエストを行う

Amazon S3 on Outposts デュアルスタックエンドポイントは、IPv6 プロトコルまたは IPv4 プロトコルを使用した S3 on Outposts バケットへのリクエストをサポートしています。S3 on Outposts の IPv6 サポートを使用すると、IPv6 ネットワーク上の S3 on Outposts API 経由でバケットとコントロールプレーンリソースにアクセスし、運用できます。

> **Note**
>
> S3 on Outposts オブジェクトアクション (`PutObject` や `GetObject` など) は、IPv6 ネットワークではサポートされていません。

IPv6 ネットワーク上で S3 on Outposts にアクセスする場合、追加料金はかかりません。S3 on Outposts の詳細については、「S3 on Outposts の料金」を参照してください。

トピック

IPv6 の使用開始

IPv6 経由で S3 on Outposts バケットにリクエストを実行するには、デュアルスタックのエンドポイントを使用する必要があります。次のセクションでは、デュアルスタックのエンドポイントを使用した IPv6 でのリクエストの実行方法について説明します。

以下は、IPv6 経由で S3 on Outposts バケットへのアクセスを試行する際の重要な考慮事項です。

- バケットにアクセスするクライアントやネットワークは、IPv6 の使用を有効にする必要があります。
- 仮想ホスト形式およびパス形式のリクエストは、IPv6 アクセスをサポートしています。詳細については、「S3 on Outposts デュアルスタックのエンドポイントの使用」を参照してください。
- AWS Identity and Access Management (IAM) ユーザーまたは S3 on Outposts バケットポリシーで、ソース IP アドレスによるフィルタリングを使用する場合、IPv6 アドレス範囲を含めるようポリシーを更新する必要があります。

> **Note**
>
> この要件は、IPv6 ネットワーク上の S3 on Outposts バケットオペレーションとコントロールプレーンリソースにのみ適用されます。Amazon S3 on Outposts オブジェクトアクションは、IPv6 ネットワークではサポートされていません。

- IPv6 を使用する場合、サーバーのアクセスログファイルは IPv6 形式の IP アドレスを出力します。IPv6 形式のリモート IP アドレスを解析できるように、 S3 on Outposts ログファイルの解析に使用する既存のツール、スクリプト、ソフトウェアを更新する必要があります。その後、更新

されたツール、スクリプト、ソフトウェアは IPv6 形式のリモート IP アドレスを正しく解析します。

デュアルスタックのエンドポイントを使用して IPv6 ネットワーク経由でリクエストを実行する

IPv6 上で S3 on Outposts API コールを使用してリクエストを行うには、AWS CLI または AWS SDK 経由でデュアルスタックのエンドポイントを使用できます。Amazon S3 コントロール API オペレーションと S3 on Outposts API オペレーションは、IPv6 プロトコルまたは IPv4 プロトコル経由で S3 on Outposts にアクセスしているかどうかに関係なく、同じように機能します。ただし、S3 on Outposts オブジェクトアクション (`PutObject` や `GetObject` など) は、IPv6 ネットワークではサポートされていません。

AWS Command Line Interface (AWS CLI) や AWS SDK を使用する場合、パラメータまたはフラグを使ってデュアルスタックのエンドポイントに変更できます。設定ファイルの S3 on Outposts エンドポイントに上書きしてデュアルスタックのエンドポイントを直接指定することもできます。

デュアルスタックのエンドポイントを使用して、次のいずれかから IPv6 経由で S3 on Outposts バケットにアクセスできます。

- AWS CLI については、「AWS CLI からのデュアルスタックのエンドポイントの使用」を参照してください。
- AWS SDK については、AWS SDK から S3 on Outposts デュアルスタックのエンドポイントを使用する を参照してください。

IAM ポリシーでの IPv6 アドレスの使用

IPv6 プロトコルを使用して S3 on Outposts バケットへのアクセスを試行する前に、IP アドレスによるフィルタリングに使用される IAM ユーザーまたは S3 on Outposts バケットポリシーが IPv6 アドレス範囲を含むように更新されているか確認する必要があります。IPv6 アドレスを処理するように IP アドレスフィルタリングポリシーが更新されていない場合、IPv6 プロトコルの使用を試行しているときに S3 on Outposts バケットにアクセスできなくなる可能性があります。

IP アドレスをフィルタリングする IAM ポリシーは、IP アドレス条件演算子を使用します。次の S3 on Outposts バケットポリシーは、IP アドレス条件演算子を使用して 54.240.143.* の許可される IP 範囲の IPv4 アドレスを識別します。この範囲外のすべての IP アドレスは S3 on Outposts バケットへのアクセスを拒否されます (`DOC-EXAMPLE-BUCKET`)。すべての IPv6 アドレスは許可範囲外であるため、このポリシーは IPv6 アドレスの `DOC-EXAMPLE-BUCKET` へのアクセスをブロックします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3outposts:*",
      "Resource": "arn:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/DOC-EXAMPLE-BUCKET/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

次の例のように、S3 on Outposts バケットポリシーの Condition エレメントを変更して、IPv4 (54.240.143.0/24) および IPv6 (2001:DB8:1234:5678::/64) アドレス範囲の両方を許可できます。例に示すように、IAM ユーザーとバケットポリシーの両方を更新するために同じタイプの Condition ブロックを使用できます。

```
       "Condition": {
         "IpAddress": {
            "aws:SourceIp": [
              "54.240.143.0/24",
               "2001:DB8:1234:5678::/64"
             ]
          }
        }
```

IPv6 を使用する前に、IPv6 アドレス範囲を許可する IP アドレスのフィルタリングに使用するすべての関連 IAM ユーザーとバケットポリシーを更新しなければなりません。既存の IPv4 アドレス範囲に加えて、IAM ポリシーと組織の IPv6 アドレス範囲を更新することをお勧めします。IPv6 および IPv4; でのアクセスを許可するバケットポリシーの例については、「特定の IP アドレスへのアクセスの制限」を参照してください。

IAM ユーザーポリシーは、https://console.aws.amazon.com/iam/ の IAM コンソールを使用して確認できます。IAM の詳細については、IAM ユーザーガイドを参照してください。S3 on Outposts バケットポリシーの編集の詳細については、「Amazon S3 on Outposts バケットのバケットポリシーを追加または編集する」を参照してください。

IP アドレス互換性のテスト

Linux、Unix インスタンス、または macOS X プラットフォームを使用している場合は、IPv6 経由でデュアルスタックのエンドポイントへのアクセスをテストできます。例えば、IPv6 経由で Amazon S3 on Outposts エンドポイントへの接続をテストするには、dig コマンドを使用します。

```
dig s3-outposts.us-west-2.api.aws AAAA +short
```

IPv6 ネットワーク上にデュアルスタックのエンドポイントが適切に設定されている場合、dig コマンドは接続されている IPv6 アドレスを返します。例:

```
dig s3-outposts.us-west-2.api.aws AAAA +short

2600:1f14:2588:4800:b3a9:1460:159f:ebce

2600:1f14:2588:4802:6df6:c1fd:ef8a:fc76

2600:1f14:2588:4801:d802:8ccf:4e04:817
```

AWS PrivateLink での IPv6 の使用

S3 on Outposts は、AWS PrivateLink サービスとエンドポイントの IPv6 プロトコルをサポートしています。IPv6 プロトコルの AWS PrivateLink のサポートにより、オンプレミスまたは他のプライベート接続から、IPv6 ネットワーク経由で VPC 内のサービスエンドポイントに接続できます。S3 on Outposts の AWS PrivateLink の IPv6 サポートにより、AWS PrivateLink をデュアルスタックのエンドポイントと統合することもできます。AWS PrivateLink で IPv6 を有効にする手順については、「Expedite your IPv6 adoption with AWS PrivateLink services and endpoints」を参照してください。

> **Note**
>
> サポートされている IP アドレスタイプを IPv4 から IPv6 に更新するには、「AWS PrivateLink ユーザーガイド」の「サポートされている IP アドレスのタイプを変更する」を参照してください。

AWS PrivateLink での IPv6 の使用

IPv6 で AWS PrivateLink を使用している場合は、IPv6 またはデュアルスタックの VPC インターフェイスエンドポイントを作成する必要があります。AWS Management Console を使用して VPC

エンドポイントを作成する一般的な手順については、「AWS PrivateLink ユーザーガイド」の「インターフェイス VPC エンドポイントを使用して AWS のサービスにアクセスする」を参照してください。

AWS Management Console

次の手順を使用して、S3 on Outposts に接続するインターフェイス VPC エンドポイントを作成します。

1. AWS Management Console にサインインして、VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. ナビゲーションペインで、[エンドポイント] を選択します。
3. [エンドポイントの作成] を選択します。
4. [Service category] (サービスカテゴリ) で、[AWS services] (のサービス) を選択します。
5. [サービス名] で、S3 on Outposts サービス (com.amazonaws.us-east-1.s3-outposts) を選択します。
6. [VPC] で、S3 on Outposts にアクセスする VPC を選択します。
7. [サブネット] で、S3 on Outposts にアクセスするアベイラビリティゾーンごとにサブネットを 1 つ選択します。同じアベイラビリティーゾーンから複数のサブネットを選択することはできません。選択したサブネットごとに、新しいエンドポイントネットワークインターフェイスが作成されます。デフォルトでは、サブネットの IP アドレス範囲の IP アドレスがエンドポイントネットワークインターフェイスに割り当てられます。エンドポイントネットワークインターフェースの IP アドレスを指定するには、[IP アドレスの指定] を選択し、サブネットアドレス範囲の IPv6 アドレスを入力します。
8. [IP アドレスタイプ] で、[デュアルスタック]を選択します。IPv4 と IPv6 の両方のアドレスをエンドポイントのネットワークインターフェイスに割り当てます。このオプションは、選択したすべてのサブネットに IPv4 と IPv6 の両方のアドレス範囲がある場合にのみサポートされます。
9. [セキュリティグループ] で、VPC エンドポイントのエンドポイントネットワークインターフェイスに関連付けるセキュリティグループを選択します。デフォルトでは、デフォルトのセキュリティグループが VPC に関連付けられます。
10. [ポリシー] では、[フルアクセス] を選択して、VPC エンドポイントのすべてのリソースに対するすべてのプリンシパルによるすべてのオペレーションを許可します。それ以外の場合は、[カスタム] を選択して、プリンシパルが VPC エンドポイントを介してリソースに対してアクションを実行するために必要なアクセス許可を制御する VPC エンドポイントポリシーをアタッチします。このオプションは、サービスが VPC エンドポイントポリシーをサポー

トしている場合にのみ使用できます。詳細については、「エンドポイントポリシー」を参照してください。

11. (オプション) タグを追加するには、[新しいタグを追加] を選択し、そのタグのキーと値を入力します。
12. [エンドポイントの作成] を選択します。

Example – S3 on Outposts バケットポリシー

S3 on Outposts が VPC エンドポイントとやり取りできるようにするには、S3 on Outposts ポリシーを次のように更新します。

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3-outposts:*",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

AWS CLI

> **Note**
>
> VPC エンドポイントで IPv6 ネットワークを有効にするには、S3 on Outposts のSupportedIpAddressTypeフィルターに IPv6 を設定する必要があります。

次の例では、create-vpc-endpoint コマンドを使用して新しいデュアルスタックインターフェイスエンドポイントを作成します。

```
aws ec2 create-vpc-endpoint \
--vpc-id vpc-12345678 \
--vpc-endpoint-type Interface \
--service-name com.amazonaws.us-east-1.s3-outposts \
--subnet-id subnet-12345678 \
--security-group-id sg-12345678 \
```

```
--ip-address-type dualstack \
--dns-options "DnsRecordIpType=dualstack"
```

AWS PrivateLink サービス設定によっては、新しく作成されたエンドポイント接続を使用する前に、VPC エンドポイントサービスプロバイダーによる承認が必要になる場合があります。詳細については、「AWS PrivateLink ユーザーガイド」の「エンドポイントの接続リクエストを承諾または拒否する」を参照してください。

次の例では、modify-vpc-endpoint コマンドを使用して、IPv 専用 VPC エンドポイントをデュアルスタックのエンドポイントに更新します。デュアルスタックのエンドポイントにより、IPv4 ネットワークと IPv6 ネットワークの両方にアクセスできるようになります。

```
aws ec2 modify-vpc-endpoint \
--vpc-endpoint-id vpce-12345678 \
--add-subnet-ids subnet-12345678 \
--remove-subnet-ids subnet-12345678 \
--ip-address-type dualstack \
--dns-options "DnsRecordIpType=dualstack"
```

AWS PrivateLink で IPv6 を有効にする方法の詳細については、「Expedite your IPv6 adoption with AWS PrivateLink services and endpoints」を参照してください。

S3 on Outposts デュアルスタックのエンドポイントの使用

S3 on Outposts デュアルスタックエンドポイントは、IPv6 および IPv4 を使用した S3 on Outposts バケットへのリクエストをサポートしています。このセクションは、S3 on Outposts デュアルスタックのエンドポイントを使用する方法を説明します。

トピック

- S3 on Outposts デュアルスタックのエンドポイント
- AWS CLI からのデュアルスタックのエンドポイントの使用
- AWS SDK から S3 on Outposts デュアルスタックのエンドポイントを使用する

S3 on Outposts デュアルスタックのエンドポイント

デュアルスタックのエンドポイントにリクエストを行うと、S3 on Outposts バケット URL は IPv6 または IPv4 アドレスに解決されます。IPv6 での S3 on Outposts バケットへのアクセスに関する詳細は、「IPv6 経由で S3 on Outposts へのリクエストを行う」を参照してください。

デュアルスタックのエンドポイントを介して S3 on Outposts バケットにアクセスするには、パス形式のエンドポイント名を使用します。S3 on Outposts はリージョンのデュアルスタックエンドポイント名のみをサポートしており、名前の一部としてリージョンを指定する必要があります。

デュアルスタックパス形式の FIPS エンドポイントの場合は、次の命名規則を使用します。

```
s3-outposts-fips.region.api.aws
```

デュアルスタックの非 FIPS エンドポイントの場合は、次の命名規則を使用します。

```
s3-outposts.region.api.aws
```

> **Note**
>
> 仮想ホスト形式のエンドポイント名は、S3 on Outposts ではサポートされていません。

AWS CLI からのデュアルスタックのエンドポイントの使用

このセクションでは、デュアルスタックのエンドポイントへリクエストするのに使用される AWS CLI コマンドの例を示します。AWS CLI をセットアップする手順については、「AWS CLI および SDK for Java の使用開始」を参照してください。

AWS Config ファイルのプロファイル内で設定値 `use_dualstack_endpoint` を `true` に設定すると、`s3` と `s3api` AWS CLI コマンドによるすべての Amazon S3 リクエストが、指定されたリージョンのデュアルスタックエンドポイントに転送されます。`--region` オプションを使用して設定ファイルまたはコマンドでリージョンを指定します。

AWS CLI でデュアルスタックのエンドポイントを使用する場合、`path` のアドレス形式のみがサポートされます。設定ファイルで設定されたアドレス形式により、バケット名がホスト名の一部か URL の一部かが決まります。詳細については、AWS CLI ユーザーガイドの s3outposts を参照してください。

AWS CLI 経由でデュアルスタックのエンドポイントを使用するには、`s3control` コマンドまたは `s3outposts` コマンドの `http://s3.dualstack.region.amazonaws.com` または `https://s3-outposts-fips.region.api.aws` のエンドポイントを指定して、`--endpoint-url` パラメータを使用します。

例:

```
$  aws s3control list-regional-buckets --endpoint-url https://s3-
outposts.region.api.aws
```

AWS SDK から S3 on Outposts デュアルスタックのエンドポイントを使用する

このセクションでは、AWS SDK を使用してデュアルスタックのエンドポイントにアクセスする方法の例を示します。

AWS SDK for Java 2.x のデュアルスタックのエンドポイントの例

次の例では、AWS SDK for Java 2.x を使用して S3 on Outposts クライアントを作成する際に、S3ControlClient クラスと S3OutpostsClient クラスを使用してデュアルスタックエンドポイントを有効化する方法を示しています。Amazon S3 on Outposts の動作する Java の例の作成とテストに関する手順については、「AWS CLI および SDK for Java の使用開始」を参照してください。

Example – デュアルスタックのエンドポイントを有効にして **S3ControlClient** クラスを作成する

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListRegionalBucketsRequest;
import software.amazon.awssdk.services.s3control.model.ListRegionalBucketsResponse;
import software.amazon.awssdk.services.s3control.model.S3ControlException;


public class DualStackEndpointsExample1 {

    public static void main(String[] args) {
        Region clientRegion = Region.of("us-east-1");
        String accountId = "111122223333";
        String navyId = "9876543210";

        try {
            // Create an S3ControlClient with dual-stack endpoints enabled.
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                                                             .region(clientRegion)
                                                             .dualstackEnabled(true)
                                                             .build();
            ListRegionalBucketsRequest listRegionalBucketsRequest =
 ListRegionalBucketsRequest.builder()
```

```
        .accountId(accountId)

        .outpostId(navyId)

        .build();

            ListRegionalBucketsResponse listBuckets =
 s3ControlClient.listRegionalBuckets(listRegionalBucketsRequest);
            System.out.printf("ListRegionalBuckets Response: %s%n",
 listBuckets.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 on Outposts
 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        }
        catch (S3ControlException e) {
            // Unknown exceptions will be thrown as an instance of this type.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 on Outposts couldn't be contacted for a response, or the
 client
            // couldn't parse the response from Amazon S3 on Outposts.
            e.printStackTrace();
        }
    }
}
```

Example – デュアルスタックのエンドポイントを有効にして **S3OutpostsClient** を作成する

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3outposts.S3OutpostsClient;
import software.amazon.awssdk.services.s3outposts.model.ListEndpointsRequest;
import software.amazon.awssdk.services.s3outposts.model.ListEndpointsResponse;
import software.amazon.awssdk.services.s3outposts.model.S3OutpostsException;


public class DualStackEndpointsExample2 {

    public static void main(String[] args) {
```

```
        Region clientRegion = Region.of("us-east-1");

        try {
            // Create an S3OutpostsClient with dual-stack endpoints enabled.
            S3OutpostsClient s3OutpostsClient = S3OutpostsClient.builder()
                                                                .region(clientRegion)
                                                                .dualstackEnabled(true)
                                                                .build();
            ListEndpointsRequest listEndpointsRequest =
 ListEndpointsRequest.builder().build();

            ListEndpointsResponse listEndpoints =
 s3OutpostsClient.listEndpoints(listEndpointsRequest);
            System.out.printf("ListEndpoints Response: %s%n",
 listEndpoints.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 on Outposts
 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        }
        catch (S3OutpostsException e) {
            // Unknown exceptions will be thrown as an instance of this type.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 on Outposts couldn't be contacted for a response, or the
 client
            // couldn't parse the response from Amazon S3 on Outposts.
            e.printStackTrace();
        }
    }
}
```

Windows で AWS SDK for Java 2.x を使用している場合は、必要に応じて、次の Java 仮想マシン (JVM) のプロパティを設定します。

```
java.net.preferIPv6Addresses=true
```

AWS SDK を使用した Amazon S3 のコード例

次のコード例は、AWS ソフトウェア開発キット (SDK) による Amazon S3 の使用方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

クロスサービスの例は、複数の AWS のサービス で動作するサンプルアプリケーションです。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

開始方法

Hello Amazon S3

次のコード例は、Amazon S3 の使用を開始する方法を示しています。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)
```

```
# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_s3.cpp ソースファイルのコード。

```
#include <aws/core/Aws.h>
```

```
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 *  A "Hello S3" starter application which initializes an Amazon Simple Storage
 Service (Amazon S3) client
 *  and lists the Amazon S3 buckets in the selected region.
 *
 *  main function
 *
 *  Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        // You don't normally have to test that you are authenticated. But the
 S3 service permits anonymous requests, thus the s3Client will return "success"
 and 0 buckets even if you are unauthenticated, which can be confusing to a new
 user.
        auto provider =
 Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-tag");
        auto creds = provider->GetAWSCredentials();
        if (creds.IsEmpty()) {
            std::cerr << "Failed authentication" << std::endl;
        }

        Aws::S3::S3Client s3Client(clientConfig);
        auto outcome = s3Client.ListBuckets();

        if (!outcome.IsSuccess()) {
```

```
            std::cerr << "Failed with error: " << outcome.GetError() <<
 std::endl;
            result = 1;
        } else {
            std::cout << "Found " << outcome.GetResult().GetBuckets().size()
                      << " buckets\n";
            for (auto &bucket: outcome.GetResult().GetBuckets()) {
                std::cout << bucket.GetName() << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「ListBuckets」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)
```

```
// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 s3Client := s3.NewFromConfig(sdkConfig)
 count := 10
 fmt.Printf("Let's list up to %v buckets for your account.\n", count)
 result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
 if err != nil {
  fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Buckets) == 0 {
  fmt.Println("You don't have any buckets!")
 } else {
  if count > len(result.Buckets) {
   count = len(result.Buckets)
  }
  for _, bucket := range result.Buckets[:count] {
   fmt.Printf("\t%v\n", *bucket.Name)
  }
 }
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「ListBuckets」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBuckets(s3);
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
```

```
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「ListBuckets」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";

// When no region or credentials are provided, the SDK will use the
// region and credentials from the local AWS config.
const client = new S3Client({});

export const helloS3 = async () => {
  const command = new ListBucketsCommand({});

  const { Buckets } = await client.send(command);
  console.log("Buckets: ");
  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
  return Buckets;
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「ListBuckets」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
use Aws\S3\S3Client;

$client = new S3Client(['region' => 'us-west-2']);
$results = $client->listBuckets();
var_dump($results);
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「ListBuckets」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import boto3


def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) resource and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
```

```
    s3_resource = boto3.resource("s3")
    print("Hello, Amazon S3! Let's list your buckets:")
    for bucket in s3_resource.buckets.all():
        print(f"\t{bucket.name}")


if __name__ == "__main__":
    hello_s3()
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「ListBuckets」を参照してください。

コードの例

- AWS SDK を使用した Amazon S3 のアクション
  - AWS SDK または CLI で AbortMultipartUpload を使用する
  - AWS SDK または CLI で AbortMultipartUploads を使用する
  - AWS SDK または CLI で CompleteMultipartUpload を使用する
  - AWS SDK または CLI で CopyObject を使用する
  - AWS SDK または CLI で CreateBucket を使用する
  - AWS SDK または CLI で CreateMultiRegionAccessPoint を使用する
  - AWS SDK または CLI で CreateMultipartUpload を使用する
  - AWS SDK または CLI で DeleteBucket を使用する
  - AWS SDK または CLI で DeleteBucketAnalyticsConfiguration を使用する
  - AWS SDK または CLI で DeleteBucketCors を使用する
  - AWS SDK または CLI で DeleteBucketEncryption を使用する
  - AWS SDK または CLI で DeleteBucketInventoryConfiguration を使用する
  - AWS SDK または CLI で DeleteBucketLifecycle を使用する
  - AWS SDK または CLI で DeleteBucketMetricsConfiguration を使用する
  - AWS SDK または CLI で DeleteBucketPolicy を使用する
  - AWS SDK または CLI で DeleteBucketReplication を使用する
  - AWS SDK または CLI で DeleteBucketTagging を使用する
  - AWS SDK または CLI で DeleteBucketWebsite を使用する
  - AWS SDK または CLI で DeleteObject を使用する

- AWS SDK または CLI で DeleteObjectTagging を使用する
- AWS SDK または CLI で DeleteObjects を使用する
- AWS SDK または CLI で DeletePublicAccessBlock を使用する
- AWS SDK または CLI で GetBucketAccelerateConfiguration を使用する
- AWS SDK または CLI で GetBucketAcl を使用する
- AWS SDK または CLI で GetBucketAnalyticsConfiguration を使用する
- AWS SDK または CLI で GetBucketCors を使用する
- AWS SDK または CLI で GetBucketEncryption を使用する
- AWS SDK または CLI で GetBucketInventoryConfiguration を使用する
- AWS SDK または CLI で GetBucketLifecycleConfiguration を使用する
- AWS SDK または CLI で GetBucketLocation を使用する
- AWS SDK または CLI で GetBucketLogging を使用する
- AWS SDK または CLI で GetBucketMetricsConfiguration を使用する
- AWS SDK または CLI で GetBucketNotification を使用する
- AWS SDK または CLI で GetBucketPolicy を使用する
- AWS SDK または CLI で GetBucketPolicyStatus を使用する
- AWS SDK または CLI で GetBucketReplication を使用する
- AWS SDK または CLI で GetBucketRequestPayment を使用する
- AWS SDK または CLI で GetBucketTagging を使用する
- AWS SDK または CLI で GetBucketVersioning を使用する
- AWS SDK または CLI で GetBucketWebsite を使用する
- AWS SDK または CLI で GetObject を使用する
- AWS SDK または CLI で GetObjectAcl を使用する
- AWS SDK または CLI で GetObjectLegalHold を使用する
- AWS SDK または CLI で GetObjectLockConfiguration を使用する
- AWS SDK または CLI で GetObjectRetention を使用する
- AWS SDK または CLI で GetObjectTagging を使用する
- AWS SDK または CLI で GetPublicAccessBlock を使用する
- AWS SDK または CLI で HeadBucket を使用する
- AWS SDK または CLI で HeadObject を使用する

- AWS SDK または CLI で ListBucketAnalyticsConfigurations を使用する
- AWS SDK または CLI で ListBucketInventoryConfigurations を使用する
- AWS SDK または CLI で ListBuckets を使用する
- AWS SDK または CLI で ListMultipartUploads を使用する
- AWS SDK または CLI で ListObjectVersions を使用する
- AWS SDK または CLI で ListObjects を使用する
- AWS SDK または CLI で ListObjectsV2 を使用する
- AWS SDK または CLI で PutBucketAccelerateConfiguration を使用する
- AWS SDK または CLI で PutBucketAcl を使用する
- AWS SDK または CLI で PutBucketCors を使用する
- AWS SDK または CLI で PutBucketEncryption を使用する
- AWS SDK または CLI で PutBucketLifecycleConfiguration を使用する
- AWS SDK または CLI で PutBucketLogging を使用する
- AWS SDK または CLI で PutBucketNotification を使用する
- AWS SDK または CLI で PutBucketNotificationConfiguration を使用する
- AWS SDK または CLI で PutBucketPolicy を使用する
- AWS SDK または CLI で PutBucketReplication を使用する
- AWS SDK または CLI で PutBucketRequestPayment を使用する
- AWS SDK または CLI で PutBucketTagging を使用する
- AWS SDK または CLI で PutBucketVersioning を使用する
- AWS SDK または CLI で PutBucketWebsite を使用する
- AWS SDK または CLI で PutObject を使用する
- AWS SDK または CLI で PutObjectAcl を使用する
- AWS SDK または CLI で PutObjectLegalHold を使用する
- AWS SDK または CLI で PutObjectLockConfiguration を使用する
- AWS SDK または CLI で PutObjectRetention を使用する
- AWS SDK または CLI で RestoreObject を使用する
- AWS SDK または CLI で SelectObjectContent を使用する
- AWS SDK または CLI で UploadPart を使用する

- AWS SDK を使用した Amazon S3 のシナリオ

- AWS SDK を使用して Amazon S3 の署名付き URL を作成する
- AWS SDK を使用して Amazon S3 オブジェクトを一覧表示するウェブページ
- AWS SDK を使用して Amazon S3 への不完全なマルチパートアップロードを削除する
- Amazon Simple Storage Service (Amazon S3) バケットからすべてのオブジェクトを、ローカルディレクトリにダウンロードする
- AWS SDK を使用してマルチリージョンアクセスポイントから Amazon S3 オブジェクトを取得する
- AWS SDK を使用して、Amazon S3 バケットから If-Modified-Since を指定してオブジェクトを取得する
- AWS SDK を使用して Amazon S3 バケットとオブジェクトの使用を開始する
- AWS SDK を使用して Amazon S3 オブジェクトの暗号化の使用を開始する
- AWS SDK を使用して Amazon S3 オブジェクトのタグの使用を開始する
- AWS SDK を使用して Amazon S3 オブジェクトのリーガルホールド設定を取得する
- AWS SDK を使用して Amazon S3 オブジェクトロック機能を操作する
- AWS SDK を使用して Amazon S3バケットのアクセスコントロールリスト (ACL) を管理する
- AWS SDK を使用して、バージョン管理されている Amazon S3 オブジェクトを Lambda 関数でバッチで管理する
- AWS SDK を使用して Amazon S3 の URI を解析する
- AWS SDK を使用して Amazon S3 オブジェクトのマルチパートコピーを実行する
- AWS SDK を使用して Amazon S3 オブジェクトのマルチパートアップロードを実行する
- AWS SDK を使用して Amazon S3 オブジェクトのアップロードまたはダウンロードを追跡する
- AWS SDK による単体テストと統合テストのアプローチ例
- ローカルディレクトリを Amazon Simple Storage Service (Amazon S3) バケットに、再帰的にアップロードする
- AWS SDK を使用して Amazon S3 との間で大きなファイルをアップロードまたはダウンロードする
- サイズが不明なストリームを AWS SDK を使用して Amazon S3 オブジェクトにアップロードする
- チェックサムを使用した AWS SDK での Amazon S3 オブジェクトの操作
- AWS SDK を使用して、Amazon S3 のバージョン管理されたオブジェクトを操作する
- AWS SDK を使用した Amazon S3 用のサーバーレスサンプル

- Amazon S3 トリガーから Lambda 関数を呼び出す
- AWS SDK を使用した Amazon S3 のクロスサービスの例
  - Amazon Transcribe アプリを構築する
  - AWS SDK を使用して、テキストを音声に変換し、テキストに戻す
  - ユーザーがラベルを使用して写真を管理できる写真アセット管理アプリケーションの作成
  - Amazon Textract エクスプローラーアプリケーションを作成する
  - AWS SDK を使用して、Amazon Rekognition でイメージ内の PPE を検出する
  - AWS SDK を使用して、画像から抽出されたテキスト内のエンティティを検出する
  - AWS SDK を使用してイメージ内の顔を検出する
  - AWS SDK を使用して、Amazon Rekognition で、イメージ内のオブジェクトを検出します
  - AWS SDK を使用して、Amazon Rekognition で、動画内の人やオブジェクトを検出する
  - AWS SDK を使用して、EXIF およびその他のイメージ情報を保存します
  - S3 Object Lambda でアプリケーションのデータを変換する

AWS SDK を使用した Amazon S3 のアクション

次のコード例では、AWS SDK を使用して個々の Amazon S3 アクションを実行する方法を示しています。これらの抜粋は、Amazon S3 API を呼び出し、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。完全なリストについては、「Amazon Simple Storage Service (Amazon S3) API リファレンス」を参照してください。

例

- AWS SDK または CLI で AbortMultipartUpload を使用する
- AWS SDK または CLI で AbortMultipartUploads を使用する
- AWS SDK または CLI で CompleteMultipartUpload を使用する
- AWS SDK または CLI で CopyObject を使用する
- AWS SDK または CLI で CreateBucket を使用する
- AWS SDK または CLI で CreateMultiRegionAccessPoint を使用する
- AWS SDK または CLI で CreateMultipartUpload を使用する

- AWS SDK または CLI で DeleteBucket を使用する
- AWS SDK または CLI で DeleteBucketAnalyticsConfiguration を使用する
- AWS SDK または CLI で DeleteBucketCors を使用する
- AWS SDK または CLI で DeleteBucketEncryption を使用する
- AWS SDK または CLI で DeleteBucketInventoryConfiguration を使用する
- AWS SDK または CLI で DeleteBucketLifecycle を使用する
- AWS SDK または CLI で DeleteBucketMetricsConfiguration を使用する
- AWS SDK または CLI で DeleteBucketPolicy を使用する
- AWS SDK または CLI で DeleteBucketReplication を使用する
- AWS SDK または CLI で DeleteBucketTagging を使用する
- AWS SDK または CLI で DeleteBucketWebsite を使用する
- AWS SDK または CLI で DeleteObject を使用する
- AWS SDK または CLI で DeleteObjectTagging を使用する
- AWS SDK または CLI で DeleteObjects を使用する
- AWS SDK または CLI で DeletePublicAccessBlock を使用する
- AWS SDK または CLI で GetBucketAccelerateConfiguration を使用する
- AWS SDK または CLI で GetBucketAcl を使用する
- AWS SDK または CLI で GetBucketAnalyticsConfiguration を使用する
- AWS SDK または CLI で GetBucketCors を使用する
- AWS SDK または CLI で GetBucketEncryption を使用する
- AWS SDK または CLI で GetBucketInventoryConfiguration を使用する
- AWS SDK または CLI で GetBucketLifecycleConfiguration を使用する
- AWS SDK または CLI で GetBucketLocation を使用する
- AWS SDK または CLI で GetBucketLogging を使用する
- AWS SDK または CLI で GetBucketMetricsConfiguration を使用する
- AWS SDK または CLI で GetBucketNotification を使用する
- AWS SDK または CLI で GetBucketPolicy を使用する
- AWS SDK または CLI で GetBucketPolicyStatus を使用する

- AWS SDK または CLI で GetBucketReplication を使用する
- AWS SDK または CLI で GetBucketRequestPayment を使用する
- AWS SDK または CLI で GetBucketTagging を使用する
- AWS SDK または CLI で GetBucketVersioning を使用する
- AWS SDK または CLI で GetBucketWebsite を使用する
- AWS SDK または CLI で GetObject を使用する
- AWS SDK または CLI で GetObjectAcl を使用する
- AWS SDK または CLI で GetObjectLegalHold を使用する
- AWS SDK または CLI で GetObjectLockConfiguration を使用する
- AWS SDK または CLI で GetObjectRetention を使用する
- AWS SDK または CLI で GetObjectTagging を使用する
- AWS SDK または CLI で GetPublicAccessBlock を使用する
- AWS SDK または CLI で HeadBucket を使用する
- AWS SDK または CLI で HeadObject を使用する
- AWS SDK または CLI で ListBucketAnalyticsConfigurations を使用する
- AWS SDK または CLI で ListBucketInventoryConfigurations を使用する
- AWS SDK または CLI で ListBuckets を使用する
- AWS SDK または CLI で ListMultipartUploads を使用する
- AWS SDK または CLI で ListObjectVersions を使用する
- AWS SDK または CLI で ListObjects を使用する
- AWS SDK または CLI で ListObjectsV2 を使用する
- AWS SDK または CLI で PutBucketAccelerateConfiguration を使用する
- AWS SDK または CLI で PutBucketAcl を使用する
- AWS SDK または CLI で PutBucketCors を使用する
- AWS SDK または CLI で PutBucketEncryption を使用する
- AWS SDK または CLI で PutBucketLifecycleConfiguration を使用する
- AWS SDK または CLI で PutBucketLogging を使用する
- AWS SDK または CLI で PutBucketNotification を使用する

- AWS SDK または CLI で PutBucketNotificationConfiguration を使用する
- AWS SDK または CLI で PutBucketPolicy を使用する
- AWS SDK または CLI で PutBucketReplication を使用する
- AWS SDK または CLI で PutBucketRequestPayment を使用する
- AWS SDK または CLI で PutBucketTagging を使用する
- AWS SDK または CLI で PutBucketVersioning を使用する
- AWS SDK または CLI で PutBucketWebsite を使用する
- AWS SDK または CLI で PutObject を使用する
- AWS SDK または CLI で PutObjectAcl を使用する
- AWS SDK または CLI で PutObjectLegalHold を使用する
- AWS SDK または CLI で PutObjectLockConfiguration を使用する
- AWS SDK または CLI で PutObjectRetention を使用する
- AWS SDK または CLI で RestoreObject を使用する
- AWS SDK または CLI で SelectObjectContent を使用する
- AWS SDK または CLI で UploadPart を使用する

AWS SDK または CLI で **AbortMultipartUpload** を使用する

以下のコード例は、AbortMultipartUpload の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 不完全なマルチパートアップロードを特定する

CLI

AWS CLI

指定したマルチパートアップロードを中止するには

次の abort-multipart-upload コマンドは、バケット my-bucket 内のキー multipart/01 のマルチパートアップロードを中止します。

```
aws s3api abort-multipart-upload \
    --bucket my-bucket \
    --key multipart/01 \
    --upload-id
 dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3U
```

このコマンドに必要なアップロード ID create-multipart-upload はによって出力され、list-multipart-uploads で取得することもできます。

- API の詳細については、「AWS CLI コマンドリファレンス」の「AbortMultipartUpload」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、5 日より前に作成されたマルチパートアップロードを中止します。

```
Remove-S3MultipartUpload -BucketName test-files -DaysBefore 5
```

例 2: このコマンドは、2014 年 1 月 2 日より前に作成されたマルチパートアップロードを中止します。

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "Thursday, January
 02, 2014"
```

例 3: このコマンドは、2014 年 1 月 2 日 10:45:37 より前に作成されたマルチパートアップロードを中止します。

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "2014/01/02
 10:45:37"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「AbortMultipartUpload」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **AbortMultipartUploads** を使用する

次のコード例は、AbortMultipartUploads を使用する方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;

/// <summary>
/// This example shows how to use the Amazon Simple Storage Service
/// (Amazon S3) to stop a multi-part upload process using the Amazon S3
/// TransferUtility.
/// </summary>
public class AbortMPU
{
    public static async Task Main()
    {
        string bucketName = "doc-example-bucket";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();

        await AbortMPUAsync(client, bucketName);
    }

    /// <summary>
    /// Cancels the multi-part copy process.
    /// </summary>
```

```
        /// <param name="client">The initialized client object used to create
        /// the TransferUtility object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the
        /// multi-part copy operation is in progress.</param>
        public static async Task AbortMPUAsync(IAmazonS3 client, string
bucketName)
        {
            try
            {
                var transferUtility = new TransferUtility(client);

                // Cancel all in-progress uploads initiated before the specified
date.
                await transferUtility.AbortMultipartUploadsAsync(
                    bucketName, DateTime.Now.AddDays(-7));
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「AbortMultipartUploads」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CompleteMultipartUpload** を使用する

以下のコード例は、CompleteMultipartUpload の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- マルチパートコピーを実行する
- マルチパートアップロードの実行

- チェックサムの使用

CLI

AWS CLI

次のコマンドは、バケット my-bucket 内のキー multipart/01 のマルチパートアップロードを完了します。

```
aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket my-bucket --key 'multipart/01' --upload-id
 dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3U
```

このコマンドに必要なアップロード ID create-multipart-upload はによって出力され、list-multipart-uploads で取得することもできます。

上記のコマンドのマルチパートアップロードオプションは、完全なファイルに再構成する必要があるマルチパートアップロードの部分を記述した JSON 構造を採用しています。この例では、file:// プレフィックスを使用して、mpustruct という名前のローカルフォルダーにあるファイルから JSON 構造を読み込みます。

mpustruct:

```
{
  "Parts": [
    {
      "ETag": "e868e0f4719e394144ef36531ee6824c",
      "PartNumber": 1
    },
    {
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",
      "PartNumber": 2
    },
    {
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",
      "PartNumber": 3
    }
  ]
}
```

各パートの ETag 値は、upload-part コマンドを使用してパートをアップロードするたびに出力されます。また、list-parts を呼び出して取得したり、各パートの MD5 チェックサムを取得して計算したりすることもできます。

出力:

```
{
    "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
    "Bucket": "my-bucket",
    "Location": "https://my-bucket.s3.amazonaws.com/multipart%2F01",
    "Key": "multipart/01"
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「CompleteMultipartUpload」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await
        .unwrap();
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「CompleteMultipartUpload」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CopyObject** を使用する

以下のコード例は、CopyObject の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- バケットとオブジェクトの使用を開始する
- 暗号化の開始方法

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

public class CopyObject
{
    public static async Task Main()
    {
        // Specify the AWS Region where your buckets are located if it is
        // different from the AWS Region of the default user.
        IAmazonS3 s3Client = new AmazonS3Client();

        // Remember to change these values to refer to your Amazon S3
objects.
        string sourceBucketName = "doc-example-bucket1";
        string destinationBucketName = "doc-example-bucket2";
```

```
            string sourceObjectKey = "testfile.txt";
            string destinationObjectKey = "testfilecopy.txt";

            Console.WriteLine($"Copying {sourceObjectKey} from {sourceBucketName}
 to ");
            Console.WriteLine($"{destinationBucketName} as
 {destinationObjectKey}");

            var response = await CopyingObjectAsync(
                s3Client,
                sourceObjectKey,
                destinationObjectKey,
                sourceBucketName,
                destinationBucketName);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("\nCopy complete.");
            }
        }

        /// <summary>
        /// This method calls the AWS SDK for .NET to copy an
        /// object from one Amazon S3 bucket to another.
        /// </summary>
        /// <param name="client">The Amazon S3 client object.</param>
        /// <param name="sourceKey">The name of the object to be copied.</param>
        /// <param name="destinationKey">The name under which to save the copy.</
param>
        /// <param name="sourceBucketName">The name of the Amazon S3 bucket
        /// where the file is located now.</param>
        /// <param name="destinationBucketName">The name of the Amazon S3
        /// bucket where the copy should be saved.</param>
        /// <returns>Returns a CopyObjectResponse object with the results from
        /// the async call.</returns>
        public static async Task<CopyObjectResponse> CopyingObjectAsync(
            IAmazonS3 client,
            string sourceKey,
            string destinationKey,
            string sourceBucketName,
            string destinationBucketName)
        {
            var response = new CopyObjectResponse();
            try
```

```
            {
                var request = new CopyObjectRequest
                {
                    SourceBucket = sourceBucketName,
                    SourceKey = sourceKey,
                    DestinationBucket = destinationBucketName,
                    DestinationKey = destinationKey,
                };
                response = await client.CopyObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error copying object: '{ex.Message}'");
            }

            return response;
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「CopyObject」を参照してください。

Bash

Bash スクリプトを使用した AWS CLI

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
#############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
```

```
}

################################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「CopyObject」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
 &fromBucket, const Aws::String &toBucket,
                             const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
            .WithKey(objectKey)
            .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;

    } else {
        std::cout << "Successfully copied " << objectKey << " from " <<
 fromBucket <<
                  " to " << toBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「CopyObject」を参照してください。

CLI

AWS CLI

次のコマンドは、bucket-1 から bucket-2 にオブジェクトをコピーします。

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --bucket
 bucket-2
```

出力:

```
{
    "CopyObjectResult": {
        "LastModified": "2015-11-10T01:07:25.000Z",
        "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""
    },
    "VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifMlkA"
}
```

- API の詳細については、「AWS CLI Command Reference」の「CopyObject」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
```

```
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(sourceBucket string, destinationBucket
 string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(context.TODO(), &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't copy object from %v:%v to %v:%v. Here's why: %v\n",
   sourceBucket, objectKey, destinationBucket, objectKey, err)
 }
 return err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「CopyObject」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

S3Client を使用してオブジェクトをコピーします。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <objectKey> <fromBucket> <toBucket>

                Where:
                    objectKey - The name of the object (for example, book.pdf).
                    fromBucket - The S3 bucket name that contains the object (for
 example, bucket1).
                    toBucket - The S3 bucket to copy the object to (for example,
 bucket2).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String objectKey = args[0];
        String fromBucket = args[1];
        String toBucket = args[2];
        System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
 fromBucket, toBucket);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        copyBucketObject(s3, fromBucket, objectKey, toBucket);
        s3.close();
    }
```

```
    public static String copyBucketObject(S3Client s3, String fromBucket, String
 objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
                .sourceBucket(fromBucket)
                .sourceKey(objectKey)
                .destinationBucket(toBucket)
                .destinationKey(objectKey)
                .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

S3TransferManager を使用して、1 つのバケットから別のバケットへオブジェクトをコピーします。完全なファイルとテストを表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String
 bucketName,
            String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
                .sourceBucket(bucketName)
                .sourceKey(key)
                .destinationBucket(destinationBucket)
```

```
                .destinationKey(destinationKey)
                .build();

        CopyRequest copyRequest = CopyRequest.builder()
                .copyObjectRequest(copyObjectRequest)
                .build();

        Copy copy = transferManager.copy(copyRequest);

        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「CopyObject」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトをコピーします。

```
import { S3Client, CopyObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new CopyObjectCommand({
    CopySource: "SOURCE_BUCKET/SOURCE_OBJECT_KEY",
    Bucket: "DESTINATION_BUCKET",
    Key: "NEW_OBJECT_KEY",
  });

  try {
    const response = await client.send(command);
```

```
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「CopyObject」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
```

```
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CopyObject」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトの単純なコピー。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with object copying before continuing.");
        }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「CopyObject」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、オブジェクト「sample.txt」をバケット「test-files」から同じバケットにコピーします。ただし、新しいキー「sample-copy.txt」を使用します。

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-
copy.txt
```

例 2: このコマンドは、オブジェクト「sample.txt」をバケット「test-files」からバケット「backup-files」にコピーします。ただし、キーは「sample-copy.txt」を使用します。

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-
copy.txt -DestinationBucket backup-files
```

例 3: このコマンドは、オブジェクト「sample.txt」をバケット「test-files」からダウンロードして、「local-sample.txt」という名前のローカルファイルに保存します。

```
Copy-S3Object -BucketName test-files -Key sample.txt -LocalFile local-sample.txt
```

例 4: 指定したファイルに 1 つのオブジェクトをダウンロードします。ダウンロードしたファイルは c:\downloads\data\archive.zip に保存されます。

```
Copy-S3Object -BucketName test-files -Key data/archive.zip -LocalFolder c:
\downloads
```

例 5: 指定した key prefix と一致するすべてのオブジェクトをローカルフォルダにダウンロードします。相対的なキー階層は、ダウンロードの場所全体のサブフォルダとして保存されます。

```
Copy-S3Object -BucketName test-files -KeyPrefix data -LocalFolder c:\downloads
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「CopyObject」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
 key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
 self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
```

```
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「CopyObject」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトをコピーします。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end
```

```
  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
 why: #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

オブジェクトをコピーし、宛先オブジェクトにサーバー側の暗号化を追加します。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
```

```
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
 target key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
 why: #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key,
 target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
```

```
        "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「CopyObject」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
pub async fn copy_object(
    client: &Client,
    bucket_name: &str,
    object_key: &str,
    target_key: &str,
) -> Result<CopyObjectOutput, SdkError<CopyObjectError>> {
    let mut source_bucket_and_object: String = "".to_owned();
    source_bucket_and_object.push_str(bucket_name);
    source_bucket_and_object.push('/');
    source_bucket_and_object.push_str(object_key);

    client
        .copy_object()
        .copy_source(source_bucket_and_object)
        .bucket(bucket_name)
        .key(target_key)
        .send()
        .await
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「CopyObject」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
TRY.
    lo_s3->copyobject(
      iv_bucket = iv_dest_bucket
      iv_key = iv_dest_object
      iv_copysource = |{ iv_src_bucket }/{ iv_src_object }|
    ).
    MESSAGE 'Object copied to another bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
  CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「CopyObject」を参照してください。

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        _ = try await client.copyObject(input: input)
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「CopyObject」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CreateBucket** を使用する

以下のコード例は、CreateBucket の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- バケットとオブジェクトの使用を開始する
- バージョン管理されたオブジェクトを操作する

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        /// <summary>
        /// Shows how to create a new Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket to create.</param>
        /// <returns>A boolean value representing the success or failure of
        /// the bucket creation process.</returns>
        public static async Task<bool> CreateBucketAsync(IAmazonS3 client, string
bucketName)
        {
            try
            {
                var request = new PutBucketRequest
                {
                    BucketName = bucketName,
                    UseClientRegion = true,
                };

                var response = await client.PutBucketAsync(request);
                return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error creating bucket: '{ex.Message}'");
                return false;
            }
        }
```

オブジェクトロックを有効にしてバケットを作成します。

```
    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
enableObjectLock)
    {
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「CreateBucket」を参照してください。

Bash

Bash スクリプトを使用した AWS CLI

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
```

```
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally
 unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi
```

```
  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「CreateBucket」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(

 Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                         clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
                  " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「CreateBucket」を参照してください。

CLI

AWS CLI

例 1: バケットを作成するには

次の create-bucket の例は、my-bucket という名前のバケットを作成します。

```
aws s3api create-bucket \
    --bucket my-bucket \
    --region us-east-1
```

出力:

```
{
    "Location": "/my-bucket"
}
```

詳細については、「Amazon S3 ユーザーガイド」の「バケットの作成」を参照してください。

例 2: 所有者の強制を使用してバケットを作成するには

次の create-bucket の例は、S3 オブジェクトの所有権のバケット所有者の強制設定を使用して、my-bucket という名前のバケットを作成します。

```
aws s3api create-bucket \
    --bucket my-bucket \
    --region us-east-1 \
    --object-ownership BucketOwnerEnforced
```

出力:

```
{
    "Location": "/my-bucket"
}
```

詳細については、Amazon S3 ユーザーガイドのオブジェクトの所有権のコントロールと ACL の無効化を参照してください。

例 3: ``us-east-1`` リージョンの外にバケットを作成するには

次の create-bucket の例は、eu-west-1 リージョンに my-bucket という名前のバケットを作成します。us-east-1 の外にある目的のリージョンにバケットを作成するには、適切な LocationConstraint を指定する必要があります。

```
aws s3api create-bucket \
    --bucket my-bucket \
    --region eu-west-1 \
    --create-bucket-configuration LocationConstraint=eu-west-1
```

出力:

```
{
    "Location": "http://my-bucket.s3.amazonaws.com/"
}
```

詳細については、「Amazon S3 ユーザーガイド」の「バケットの作成」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「CreateBucket」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

デフォルト設定を使用してバケットを作成します。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(name string, region string) error {
 _, err := basics.S3Client.CreateBucket(context.TODO(), &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  log.Printf("Couldn't create bucket %v in Region %v. Here's why: %v\n",
   name, region, err)
 }
 return err
}
```

オブジェクトロックを使用してバケットを作成し、作成が完了するまで待ちます。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// CreateBucketWithLock creates a new S3 bucket with optional object locking
 enabled
// and waits for the bucket to exist before returning.
```

```
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「CreateBucket」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットを作成します。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CreateBucket {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the bucket to create. The bucket
 name must be unique, or an error occurs.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Creating a bucket named %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        createBucket(s3, bucketName);
        s3.close();
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
 s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

オブジェクトロックを有効にしてバケットを作成します。

```
   // Create a new Amazon S3 bucket with object lock options.
   public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
       S3Waiter s3Waiter = getClient().waiter();
       CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
           .bucket(bucketName)
           .objectLockEnabledForBucket(enableObjectLock)
           .build();

       getClient().createBucket(bucketRequest);
       HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
           .bucket(bucketName)
           .build();

       // Wait until the bucket is created and print out the response.
       s3Waiter.waitUntilBucketExists(bucketRequestWait);
       System.out.println(bucketName + " is ready");
   }
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「CreateBucket」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットを作成します。

```
import { CreateBucketCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});
```

```
export const main = async () => {
  const command = new CreateBucketCommand({
    // The name of the bucket. Bucket names are unique and have several other
 constraints.
    // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/
bucketnamingrules.html
    Bucket: "bucket-name",
  });

  try {
    const { Location } = await client.send(command);
    console.log(`Bucket created with location ${Location}`);
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「CreateBucket」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
```

```
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateBucket」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットを作成します。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「CreateBucket」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

デフォルトの設定でバケットを作成します。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def create(self, region_override=None):
        """
        Create an Amazon S3 bucket in the default Region for the account or in
 the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this
 is
                                not specified, the Region configured in your
 shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
```

```
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
 region})

            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
 region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
            raise error
```

ライフサイクル設定を使用してバージョン対応バケットを作成します。

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
```

```
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )
```

```
    return bucket
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「CreateBucket」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                  create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
```

```
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      "None. You must create a bucket before you can get its location!"
    else
      @bucket.client.get_bucket_location(bucket:
 @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("doc-example-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「CreateBucket」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
pub async fn create_bucket(
    client: &Client,
    bucket_name: &str,
    region: &str,
) -> Result<CreateBucketOutput, SdkError<CreateBucketError>> {
    let constraint = BucketLocationConstraint::from(region);
    let cfg = CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「CreateBucket」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        lo_s3->createbucket(
            iv_bucket = iv_bucket_name
        ).
        MESSAGE 'S3 bucket created.' TYPE 'I'.
      CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
      CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「CreateBucket」を参照してください。

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public func createBucket(name: String) async throws {
        let config = S3ClientTypes.CreateBucketConfiguration(
            locationConstraint: .usEast2
        )
        let input = CreateBucketInput(
            bucket: name,
            createBucketConfiguration: config
        )
        _ = try await client.createBucket(input: input)
```

```
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「CreateBucket」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CreateMultiRegionAccessPoint** を使用する

次の例は、CreateMultiRegionAccessPoint を使用する方法を説明しています。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

us-west-2 リージョンにリクエストを送信するように S3 コントロールクライアントを設定します。

```
        suspend fun createS3ControlClient(): S3ControlClient {
            // Configure your S3ControlClient to send requests to US West
(Oregon).
            val s3Control = S3ControlClient.fromEnvironment {
                region = "us-west-2"
            }
            return s3Control
        }
```

マルチリージョンアクセスポイントを作成します。

```
    suspend fun createMrap(
```

```
        s3Control: S3ControlClient,
        accountIdParam: String,
        bucketName1: String,
        bucketName2: String,
        mrapName: String,
    ): String {
        println("Creating MRAP ...")
        val createMrapResponse: CreateMultiRegionAccessPointResponse =
            s3Control.createMultiRegionAccessPoint {
                accountId = accountIdParam
                clientToken = UUID.randomUUID().toString()
                details {
                    name = mrapName
                    regions = listOf(
                        Region {
                            bucket = bucketName1
                        },
                        Region {
                            bucket = bucketName2
                        },
                    )
                }
            }
        val requestToken: String? = createMrapResponse.requestTokenArn

        // Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
        if (requestToken != null) {
            waitForSucceededStatus(s3Control, requestToken, accountIdParam)
            println("MRAP created")
        }

        val getMrapResponse =
            s3Control.getMultiRegionAccessPoint(
                input = GetMultiRegionAccessPointRequest {
                    accountId = accountIdParam
                    name = mrapName
                },
            )
        val mrapAlias = getMrapResponse.accessPoint?.alias
        return "arn:aws:s3::$accountIdParam:accesspoint/$mrapAlias"
    }
```

マルチリージョンアクセスポイントが使用可能になるまで待ちます。

```
        suspend fun waitForSucceededStatus(
            s3Control: S3ControlClient,
            requestToken: String,
            accountIdParam: String,
            timeBetweenChecks: Duration = 1.minutes,
        ) {
            var describeResponse: DescribeMultiRegionAccessPointOperationResponse
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )

            var status: String? = describeResponse.asyncOperation?.requestStatus
            while (status != "SUCCEEDED") {
                delay(timeBetweenChecks)
                describeResponse =
    s3Control.describeMultiRegionAccessPointOperation(
                    input = DescribeMultiRegionAccessPointOperationRequest {
                        accountId = accountIdParam
                        requestTokenArn = requestToken
                    },
                )
                status = describeResponse.asyncOperation?.requestStatus
                println(status)
            }
        }
```

- 詳細については、「AWS SDK for Kotlin デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateMultiRegionAccessPoint」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CreateMultipartUpload** を使用する

以下のコード例は、CreateMultipartUpload の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- マルチパートコピーを実行する
- マルチパートアップロードの実行
- チェックサムの使用

CLI

AWS CLI

次のコマンドは、キー multipart/01 を使用して、バケット my-bucket にマルチパートアップロードを作成します。

```
aws s3api create-multipart-upload --bucket my-bucket --key 'multipart/01'
```

出力:

```
{
    "Bucket": "my-bucket",
    "UploadId":
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
    "Key": "multipart/01"
}
```

完成したファイルは、my-bucket バケット内の multipart というフォルダで 01 という名前が付けられます。upload-part コマンドで使用できるように、アップロード ID、キー、バケット名を保存します。

- API の詳細については、AWS CLI コマンドリファレンスの「CreateMultipartUpload」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await
    .unwrap();
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「CreateMultipartUpload」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucket** を使用する

以下のコード例は、DeleteBucket の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- バケットとオブジェクトの使用を開始する

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        /// <summary>
        /// Shows how to delete an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to
delete.</param>
        /// <returns>A boolean value that represents the success or failure of
        /// the delete operation.</returns>
        public static async Task<bool> DeleteBucketAsync(IAmazonS3 client, string
bucketName)
        {
            var request = new DeleteBucketRequest
            {
                BucketName = bucketName,
            };

            var response = await client.DeleteBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「DeleteBucket」を参照してください。

Bash

Bash スクリプトを使用した AWS CLI

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
```

```
  fi
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteBucket」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {

    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「DeleteBucket」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットを削除します。

```
aws s3api delete-bucket --bucket my-bucket --region us-east-1
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteBucket」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is
 returned.
func (basics BucketBasics) DeleteBucket(bucketName string) error {
```

```
 _, err := basics.S3Client.DeleteBucket(context.TODO(), &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
  log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
 }
 return err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「DeleteBucket」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucket)
                .build();

        s3.deleteBucket(deleteBucketRequest);
        s3.close();
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「DeleteBucket」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットを削除します。

```
import { DeleteBucketCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Delete a bucket.
export const main = async () => {
  const command = new DeleteBucketCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「DeleteBucket」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

空のバケットを削除します。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteBucket」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、バケット「test-files」からすべてのオブジェクトとオブジェクトバージョンを削除してから、バケットを削除します。このコマンドは、続行する前に確認を求めるプロンプトを表示します。-Force スイッチを追加すると、確認メッセージが表示されなくなります。空でないバケットは削除できないことに注意が必要です。

```
Remove-S3Bucket -BucketName test-files -DeleteBucketContent
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeleteBucket」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「DeleteBucket」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
  answer = gets.chomp.downcase
  if answer == "y"
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「DeleteBucket」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
pub async fn delete_bucket(client: &Client, bucket_name: &str) -> Result<(),
 Error> {
    client.delete_bucket().bucket(bucket_name).send().await?;
    println!("Bucket deleted");
    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「DeleteBucket」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.

        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name
        ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
```

```
    ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「DeleteBucket」を参照してください。

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    _ = try await client.deleteBucket(input: input)
}
```

- API の詳細については、AWS SDK for Swift API リファレンスの「DeleteBucket」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketAnalyticsConfiguration** を使用する

以下のコード例は、DeleteBucketAnalyticsConfiguration の使用方法を示しています。

CLI

AWS CLI

バケットの分析設定を削除するには

次の delete-bucket-analytics-configuration の例では、指定されたバケットと ID の分析設定を削除します。

```
aws s3api delete-bucket-analytics-configuration \
    --bucket my-bucket \
    --id 1
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteBucketAnalyticsConfiguration」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケット内の「testfilter」という名前の分析フィルターを削除します。

```
Remove-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId
 'testfilter'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeleteBucketAnalyticsConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketCors** を使用する

以下のコード例は、DeleteBucketCors の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        /// <summary>
        /// Deletes a CORS configuration from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to delete the CORS configuration from the bucket.</param>
        private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
        {
            DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            await client.DeleteCORSConfigurationAsync(request);
        }
```

- API の詳細については、「AWS SDK for .NET SDK for Kotlin API リファレンス」の「DeleteBucketCors」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットから Cross-Origin Resource Sharing 設定を削除します。

```
aws s3api delete-bucket-cors --bucket my-bucket
```

- API の詳細については、AWS CLI コマンドリファレンスの「DeleteBucketCors」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
```

```
        try:
            self.bucket.Cors().delete()
            logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete CORS from bucket '%s'.",
 self.bucket.name)
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「DeleteBucketCors」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
```

```
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- API の詳細については、「AWS SDK for Ruby SDK for Kotlin API リファレンス」の「DeleteBucketCors」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketEncryption** を使用する

以下のコード例は、DeleteBucketEncryption の使用方法を示しています。

CLI

AWS CLI

バケットのサーバー側の暗号化設定を削除するには

次の delete-bucket-encryption の例では、指定したバケットのサーバー側の暗号化設定を削除します。

```
aws s3api delete-bucket-encryption \
    --bucket my-bucket
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteBucketEncryption」を参照してください。

PowerShell

Tools for PowerShell

例 1: これは、指定した S3 バケットで有効になっている暗号化を無効にします。

```
Remove-S3BucketEncryption -BucketName 's3casetestbucket'
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
 target "s3casetestbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeleteBucketEncryption」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketInventoryConfiguration** を使用する

以下のコード例は、DeleteBucketInventoryConfiguration の使用方法を示しています。

CLI

AWS CLI

バケットのインベントリ設定を削除するには

次の delete-bucket-inventory-configuration の例では、指定したバケットで ID 1 を持つインベントリ設定を削除します。

```
aws s3api delete-bucket-inventory-configuration \
```

```
    --bucket my-bucket \
    --id 1
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteBucketInventoryConfiguration」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットに対応する「testInventoryName」という名前のインベントリを削除します。

```
Remove-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId
 'testInventoryName'
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeleteBucketInventoryConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketLifecycle** を使用する

以下のコード例は、DeleteBucketLifecycle の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        /// <summary>
        /// This method removes the Lifecycle configuration from the named
        /// S3 bucket.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the RemoveLifecycleConfigAsync method.</param>
        /// <param name="bucketName">A string representing the name of the
        /// S3 bucket from which the configuration will be removed.</param>
        public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client,
string bucketName)
        {
            var request = new DeleteLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「DeleteBucketLifeLifecycle」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットからライフサイクル設定を削除します。

```
aws s3api delete-bucket-lifecycle --bucket my-bucket
```

- API の詳細については、AWS CLI コマンドリファレンスの「DeleteBucketLifecycle」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_lifecycle_configuration(self):
        """
        Remove the lifecycle configuration from the specified bucket.
        """
        try:
            self.bucket.LifecycleConfiguration().delete()
            logger.info(
                "Deleted lifecycle configuration for bucket '%s'.",
 self.bucket.name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete lifecycle configuration for bucket '%s'.",
                self.bucket.name,
            )
```

```
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「DeleteBucketLifeLifecycle」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketMetricsConfiguration** を使用する

以下のコード例は、DeleteBucketMetricsConfiguration の使用方法を示しています。

CLI

AWS CLI

バケットのメトリクス設定を削除するには

次の delete-bucket-metrics-configuration の例では、指定したバケットと ID のメトリクス設定を削除します。

```
aws s3api delete-bucket-metrics-configuration \
    --bucket my-bucket \
    --id 123
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteBucketMetricsConfiguration」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケット内の「testmetrics」という名前のメトリクスフィルターを削除します。

```
Remove-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId
 'testmetrics'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeleteBucketMetricsConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketPolicy** を使用する

以下のコード例は、DeleteBucketPolicy の使用方法を示しています。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,
                                    const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
 client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucketPolicy: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
```

```
    } else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ SDK for Kotlin API リファレンス」の「DeleteBucketPolicy」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットからバケットポリシーを削除します。

```
aws s3api delete-bucket-policy --bucket my-bucket
```

- API の詳細については、AWS CLI コマンドリファレンスの「DeleteBucketPolicy」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the policy from
 (for example, bucket1).""";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    // Delete the bucket policy.
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketPolicy(delReq);
            System.out.println("Done!");
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x SDK for Kotlin API リファレンス」の「DeleteBucketPolicy」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケット ポリシーを削除します。

```
import { DeleteBucketPolicyCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// This will remove the policy from the bucket.
export const main = async () => {
  const command = new DeleteBucketPolicyCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript SDK for Kotlin API リファレンス」の「DeleteBucketPolicy」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteBucketPolicy」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットに関連付けられたバケットポリシーを削除します。

```
Remove-S3BucketPolicy -BucketName 's3testbucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeleteBucketPolicy」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception(
                "Couldn't delete policy for bucket '%s'.", self.bucket.name
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「DeleteBucketPolicy」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's
 why: #{e.message}"
    false
  end

end
```

- API の詳細については、「AWS SDK for Ruby SDK for Kotlin API リファレンス」の「DeleteBucketPolicy」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketReplication** を使用する

以下のコード例は、DeleteBucketReplication の使用方法を示しています。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットからレプリケーションの設定を削除します。

```
aws s3api delete-bucket-replication --bucket my-bucket
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteBucketReplication」を参照してください。

PowerShell

Tools for PowerShell

例 1: 「mybucket」という名前のバケットに関連付けられているレプリケーションの設定を削除します。なお、このオペレーションには s3:DeleteReplicationConfiguration アクションに対するアクセス許可が必要です。オペレーションを続行する前に確認を求めるプロンプトが表示されます。プロンプトを表示しないようにするには、-Force スイッチを使用します。

```
Remove-S3BucketReplication -BucketName mybucket
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeleteBucketReplication」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketTagging** を使用する

以下のコード例は、DeleteBucketTagging の使用方法を示しています。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットからタグの設定を削除します。

```
aws s3api delete-bucket-tagging --bucket my-bucket
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteBucketTagging」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットに関連付けられたすべてのタグを削除します。

```
Remove-S3BucketTagging -BucketName 's3testbucket'
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeleteBucketTagging」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketWebsite** を使用する

以下のコード例は、DeleteBucketWebsite の使用方法を示しています。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
            client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「DeleteBucketWebsite」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットからウェブサイト設定を削除します。

```
aws s3api delete-bucket-website --bucket my-bucket
```

- API の詳細については、AWS CLI コマンドリファレンスの「DeleteBucketWebsite」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <bucketName>
```

```
                Where:
                    bucketName - The Amazon S3 bucket to delete the website
 configuration from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket:
 %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName)
 {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「DeleteBucketWebsite」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットからウェブサイト設定を削除します。

```
import { DeleteBucketWebsiteCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Disable static website hosting on the bucket.
export const main = async () => {
  const command = new DeleteBucketWebsiteCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「DeleteBucketWebsite」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットの静的ウェブサイトホスティングのプロパティを無効にします。

```
Remove-S3BucketWebsite -BucketName 's3testbucket'
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeleteBucketWebsite」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteObject** を使用する

以下のコード例は、DeleteObject の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- バージョン管理されたオブジェクトを操作する

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バージョニング非対応の S3 バケットからオブジェクトを削除します。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete an object from a non-versioned Amazon
    /// Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class DeleteObject
    {
        /// <summary>
        /// The Main method initializes the necessary variables and then calls
        /// the DeleteObjectNonVersionedBucketAsync method to delete the object
        /// named by the keyName parameter.
        /// </summary>
        public static async Task Main()
        {
            const string bucketName = "doc-example-bucket";
            const string keyName = "testfile.txt";

            // If the Amazon S3 bucket is located in an AWS Region other than the
            // Region of the default account, define the AWS Region for the
            // Amazon S3 bucket in your call to the AmazonS3Client constructor.
            // For example RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();
            await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
        }

        /// <summary>
```

```
        /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
        /// desired object from the named bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to delete
        /// an object from an Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the bucket from which the
        /// object will be deleted.</param>
        /// <param name="keyName">The name of the object to delete.</param>
        public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                Console.WriteLine($"Deleting object: {keyName}");
                await client.DeleteObjectAsync(deleteObjectRequest);
                Console.WriteLine($"Object: {keyName} deleted from
{bucketName}.");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error encountered on server.
Message:'{ex.Message}' when deleting an object.");
            }
        }
    }
```

バージョニングされた S3 バケットからオブジェクトを削除します。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example creates an object in an Amazon Simple Storage Service
```

```
    /// (Amazon S3) bucket and then deletes the object version that was
    /// created.
    /// </summary>
    public class DeleteObjectVersion
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "verstioned-object.txt";

            // If the AWS Region of the default user is different from the AWS
            // Region of the Amazon S3 bucket, pass the AWS Region of the
            // bucket region to the Amazon S3 client object's constructor.
            // Define it like this:
            //      RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 client = new AmazonS3Client();

            await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// This method creates and then deletes a versioned object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// create and delete the object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// object will be created and deleted.</param>
        /// <param name="keyName">The key name of the object to create.</param>
        public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(client, bucketName,
keyName);

                // Delete the object by specifying an object key and a version
ID.
                DeleteObjectRequest request = new DeleteObjectRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID,
```

```
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// This method is used to create the temporary Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object which will be
 used
        /// to create the temporary Amazon S3 object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// will be created.</param>
        /// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
        /// <returns>The Version ID of the created object.</returns>
        public static async Task<string> PutAnObject(IAmazonS3 client, string
 bucketName, string objectKey)
        {
            PutObjectRequest request = new PutObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!",
            };

            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「DeleteObject」を参照してください。

Bash

Bash スクリプトを使用した AWS CLI

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_item_in_bucket
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - The key (file name) in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_item_in_bucket() {
  local bucket_name=$1
  local key=$2
  local response

  response=$(aws s3api delete-object \
    --bucket "$bucket_name" \
    --key "$key")

  # shellcheck disable=SC2181
```

```
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteObject」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                              const Aws::String &fromBucket,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
            .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
```

```
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「DeleteObject」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットから test.txt という名前のオブジェクトを削除します。

```
aws s3api delete-object --bucket my-bucket --key test.txt
```

バケットバージョニングが有効になっている場合、出力には削除マーカーのバージョン ID が含まれます。

```
{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tNDlWlGq",
  "DeleteMarker": true
}
```

オブジェクトの削除の詳細については、「Amazon S3 ユーザーガイド」の「オブジェクトの削除」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteObject」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
 string, versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
    }
   }
  }
```

```
 } else {
  deleted = true
 }
 return deleted, err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「DeleteObject」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトを削除します。

```
import { DeleteObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new DeleteObjectCommand({
    Bucket: "test-bucket",
    Key: "test-key.txt",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「DeleteObject」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトを削除します。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def delete(self):
        """
        Deletes the object.
        """
        try:
            self.object.delete()
            self.object.wait_until_not_exists()
            logger.info(
                "Deleted object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
```

```
                "Couldn't delete object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
```

オブジェクトの新しいバージョンを削除して、オブジェクトを以前のバージョンにロールバックします。

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
```

```
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

オブジェクトのアクティブな削除マーカーを削除して、削除されたオブジェクトを復活させます。

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest
 version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
```

```
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.",
 object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

S3 オブジェクトから削除マーカーを削除する Lambda ハンドラを作成します。このハンドラを使用して、バージョン管理されたバケット内の余分な削除マーカーを効率的にクリーンアップできます。

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
```

```
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of
the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a
delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
marker."
            )
```

```
            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]
["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
 obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
 retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
```

```
            logger.exception(error)
        finally:
            results.append(
                {
                    "taskId": task_id,
                    "resultCode": result_code,
                    "resultString": result_string,
                }
            )
        return {
            "invocationSchemaVersion": invocation_schema_version,
            "treatMissingKeysAs": "PermanentFailure",
            "invocationId": invocation_id,
            "results": results,
        }
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「DeleteObject」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
async fn remove_object(client: &Client, bucket: &str, key: &str) -> Result<(),
 Error> {
    client
        .delete_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await?;
```

```
    println!("Object deleted.");

    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「DeleteObject」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_object_key
        ).
        MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「DeleteObject」を参照してください。

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    } catch {
        throw error
    }
}
```

- API の詳細については、AWS SDK for Swift API リファレンスの「DeleteObject」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteObjectTagging** を使用する

以下のコード例は、DeleteObjectTagging の使用方法を示しています。

CLI

AWS CLI

オブジェクトのタグセットを削除するには

次の delete-object-tagging の例では、指定したキーを持つタグをオブジェクト doc1.rtf から削除します。

```
aws s3api delete-object-tagging \
    --bucket my-bucket \
    --key doc1.rtf
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteObjectTagging」を参照してください。

PowerShell

Tools for PowerShell

このコマンドは、指定した S3 バケット内のキー「testfile.txt」を持つオブジェクトに関連付けられたすべてのタグを削除します。

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 's3testbucket' -Select
 '^Key'
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
 "testfile.txt".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
testfile.txt
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeleteObjectTagging」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteObjects** を使用する

以下のコード例は、DeleteObjects の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- バケットとオブジェクトの使用を開始する

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

S3 バケットからすべてのオブジェクトを削除します。

```
        /// <summary>
        /// Delete all of the objects stored in an existing Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket from which the
        /// contents will be deleted.</param>
        /// <returns>A boolean value that represents the success or failure of
        /// deleting all of the objects in the bucket.</returns>
        public static async Task<bool> DeleteBucketContentsAsync(IAmazonS3
client, string bucketName)
        {
            // Iterate over the contents of the bucket and delete all objects.
            var request = new ListObjectsV2Request
            {
                BucketName = bucketName,
```

```
            };

            try
            {
                ListObjectsV2Response response;

                do
                {
                    response = await client.ListObjectsV2Async(request);
                    response.S3Objects
                        .ForEach(async obj => await
client.DeleteObjectAsync(bucketName, obj.Key));

                    // If the response is truncated, set the request
ContinuationToken
                    // from the NextContinuationToken property of the response.
                    request.ContinuationToken = response.NextContinuationToken;
                }
                while (response.IsTruncated);

                return true;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error deleting objects: {ex.Message}");
                return false;
            }
        }
```

バージョニング非対応の S3 バケットから複数のオブジェクトを削除します。

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete multiple objects from an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
```

```
   public class DeleteMultipleObjects
   {
       /// <summary>
       /// The Main method initializes the Amazon S3 client and the name of
       /// the bucket and then passes those values to MultiObjectDeleteAsync.
       /// </summary>
       public static async Task Main()
       {
           const string bucketName = "doc-example-bucket";

           // If the Amazon S3 bucket from which you wish to delete objects is
not
           // located in the same AWS Region as the default user, define the
           // AWS Region for the Amazon S3 bucket as a parameter to the client
           // constructor.
           IAmazonS3 s3Client = new AmazonS3Client();

           await MultiObjectDeleteAsync(s3Client, bucketName);
       }

       /// <summary>
       /// This method uses the passed Amazon S3 client to first create and then
       /// delete three files from the named bucket.
       /// </summary>
       /// <param name="client">The initialized Amazon S3 client object used to
call
       /// Amazon S3 methods.</param>
       /// <param name="bucketName">The name of the Amazon S3 bucket where
objects
       /// will be created and then deleted.</param>
       public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
bucketName)
       {
           // Create three sample objects which we will then delete.
           var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);

           // Now perform the multi-object delete, passing the key names and
           // version IDs. Since we are working with a non-versioned bucket,
           // the object keys collection includes null version IDs.
           DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest
           {
               BucketName = bucketName,
               Objects = keysAndVersions,
```

```
            };

            // You can add a specific object key to the delete request using the
            // AddKey method of the multiObjectDeleteRequest.
            try
            {
                DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException e)
            {
                PrintDeletionErrorStatus(e);
            }
        }

        /// <summary>
        /// Prints the list of errors raised by the call to DeleteObjectsAsync.
        /// </summary>
        /// <param name="ex">A collection of exceptions returned by the call to
        /// DeleteObjectsAsync.</param>
        public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
        {
            DeleteObjectsResponse errorResponse = ex.Response;
            Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

            Console.WriteLine($"Successfully deleted
{errorResponse.DeletedObjects.Count}.");
            Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");

            Console.WriteLine("Printing error data...");
            foreach (DeleteError deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// This method creates simple text file objects that can be used in
        /// the delete method.
        /// </summary>
```

```
        /// <param name="client">The Amazon S3 client used to call
PutObjectAsync.</param>
        /// <param name="number">The number of objects to create.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created.</param>
        /// <returns>A list of keys (object keys) and versions that the calling
        /// method will use to delete the newly created files.</returns>
        public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, int number, string bucketName)
        {
            List<KeyVersion> keys = new List<KeyVersion>();
            for (int i = 0; i < number; i++)
            {
                string key = "ExampleObject-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                PutObjectResponse response = await
client.PutObjectAsync(request);

                // For non-versioned bucket operations, we only need the
                // object key.
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                };
                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

バージョニングされた S3 バケットから複数のオブジェクトを削除します。

```
    using System;
    using System.Collections.Generic;
```

```
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete objects in a version-enabled Amazon
    /// Simple StorageService (Amazon S3) bucket.
    /// </summary>
    public class DeleteMultipleObjects
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";

            // If the AWS Region for your Amazon S3 bucket is different from
            // the AWS Region of the default user, define the AWS Region for
            // the Amazon S3 bucket and pass it to the client constructor
            // like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 s3Client;

            s3Client = new AmazonS3Client();
            await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
        }

        /// <summary>
        /// This method removes multiple versions and objects from a
        /// version-enabled Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
        {
            // Delete objects (specifying object version in the request).
            await DeleteObjectVersionsAsync(client, bucketName);

            // Delete objects (without specifying object version in the request).
```

```
            var deletedObjects = await DeleteObjectsAsync(client, bucketName);

            // Additional exercise - remove the delete markers Amazon S3 returned
from
            // the preceding response. This results in the objects reappearing
            // in the bucket (you can verify the appearance/disappearance of
            // objects in the console).
            await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
        }

        /// <summary>
        /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
        /// them again. The method returns a list of the Amazon S3 objects
deleted.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created and then deleted.</param>
        /// <returns>A list of DeletedObjects.</returns>
        public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

            // Delete objects using only keys. Amazon S3 creates a delete marker
and
            // returns its version ID in the response.
            List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
            return deletedObjects;
        }

        /// <summary>
        /// This method creates several temporary objects and then deletes them.
        /// </summary>
        /// <param name="client">The S3 client.</param>
        /// <param name="bucketName">Name of the bucket.</param>
        /// <returns>Async task.</returns>
        public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
```

```
        {
            // Upload the sample objects.
            var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

            // Delete the specific object versions.
            await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
        }

        /// <summary>
        /// Displays the list of information about deleted files to the console.
        /// </summary>
        /// <param name="e">Error information from the delete process.</param>
        private static void DisplayDeletionErrors(DeleteObjectsException e)
        {
            var errorResponse = e.Response;
            Console.WriteLine($"No. of objects successfully deleted =
 {errorResponse.DeletedObjects.Count}");
            Console.WriteLine($"No. of objects failed to delete =
 {errorResponse.DeleteErrors.Count}");
            Console.WriteLine("Printing error data...");
            foreach (var deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
 {deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// Delete multiple objects from a version-enabled bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        private static async Task VersionedDeleteAsync(IAmazonS3 client, string
 bucketName, List<KeyVersion> keys)
        {
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
```

```
                Objects = keys, // This includes the object keys and specific
 version IDs.
            };

            try
            {
                Console.WriteLine("Executing VersionedDelete...");
                DeleteObjectsResponse response = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted all the
 {response.DeletedObjects.Count} items");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Deletes multiple objects from a non-versioned Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        /// <returns>A list of the deleted objects.</returns>
        private static async Task<List<DeletedObject>>
 NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
 keys)
        {
            // Create a request that includes only the object key names.
            DeleteObjectsRequest multiObjectDeleteRequest = new
 DeleteObjectsRequest();
            multiObjectDeleteRequest.BucketName = bucketName;

            foreach (var key in keys)
            {
                multiObjectDeleteRequest.AddKey(key.Key);
            }
```

```
            // Execute DeleteObjectsAsync.
            // The DeleteObjectsAsync method adds a delete marker for each
            // object deleted. You can verify that the objects were removed
            // using the Amazon S3 console.
            DeleteObjectsResponse response;
            try
            {
                Console.WriteLine("Executing NonVersionedDelete...");
                response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
                throw; // Some deletions failed. Investigate before continuing.
            }

            // This response contains the DeletedObjects list which we use to
delete the delete markers.
            return response.DeletedObjects;
        }

        /// <summary>
        /// Deletes the markers left after deleting the temporary objects.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="deletedObjects">A list of the objects that were
deleted.</param>
        private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
        {
            var keyVersionList = new List<KeyVersion>();

            foreach (var deletedObject in deletedObjects)
            {
                KeyVersion keyVersion = new KeyVersion
                {
```

```
                    Key = deletedObject.Key,
                    VersionId = deletedObject.DeleteMarkerVersionId,
                };
                keyVersionList.Add(keyVersion);
            }

            // Create another request to delete the delete markers.
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keyVersionList,
            };

            // Now, delete the delete marker to bring your objects back to the
 bucket.
            try
            {
                Console.WriteLine("Removing the delete markers .....");
                var deleteObjectResponse = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted the
 {deleteObjectResponse.DeletedObjects.Count} delete markers");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Create temporary Amazon S3 objects to show how object deletion wors
 in an
        /// Amazon S3 bucket with versioning enabled.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// PutObjectAsync to create temporary objects for the example.</param>
        /// <param name="bucketName">A string representing the name of the S3
        /// bucket where we will create the temporary objects.</param>
        /// <param name="number">The number of temporary objects to create.</
param>
        /// <returns>A list of the KeyVersion objects.</returns>
        private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
 client, string bucketName, int number)
```

```
        {
            var keys = new List<KeyVersion>();

            for (var i = 0; i < number; i++)
            {
                string key = "ObjectToDelete-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                var response = await client.PutObjectAsync(request);
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                    VersionId = response.VersionId,
                };

                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「DeleteObjects」を参照してください。

Bash

Bash スクリプトを使用した AWS CLI

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
```

```
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteObjects」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {

 deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
            client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
```

```
        std::cerr << "Error deleting objects. " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }

        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「DeleteObjects」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットからオブジェクトを削除します。

```
aws s3api delete-objects --bucket my-bucket --delete file://delete.json
```

delete.json は、現在のディレクトリにある JSON ドキュメントで、削除するオブジェクトを指定します。

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

出力:

```
{
    "Deleted": [
        {
            "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUPO1koHwzU",
            "Key": "test1.txt",
            "DeleteMarker": true
        }
    ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteObjects」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }
```

```
 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  } else if len(delOut.Errors) > 0 {
   for _, outErr := range delOut.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
   err = fmt.Errorf("%s", *delOut.Errors[0].Message)
  }
 }
 return err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「DeleteObjects」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucketName>

                Where:
                   bucketName - the Amazon S3 bucket name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specfied Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
        ObjectIdentifier objectId;

        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                    .key(keyName)
                    .build();

            putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            s3.putObject(putOb, RequestBody.fromString(keyName));
            keys.add(objectId);
        }

        System.out.println(keys.size() + " objects successfully created.");

        // Delete multiple objects in one request.
        Delete del = Delete.builder()
                .objects(keys)
                .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
```

```
                    .bucket(bucketName)
                    .delete(del)
                    .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「DeleteObjects」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

複数のオブジェクトを削除します。

```
import { DeleteObjectsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new DeleteObjectsCommand({
    Bucket: "test-bucket",
    Delete: {
      Objects: [{ Key: "object1.txt" }, { Key: "object2.txt" }],
    },
  });
```

```
  try {
    const { Deleted } = await client.send(command);
    console.log(
      `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
 objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
  } catch (err) {
    console.error(err);
  }
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「DeleteObjects」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
```

```
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteObjects」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

キーのリストからオブジェクトのセットを削除します。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
```

```
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (count($check) <= 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteObjects」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、バケット「test-files」からオブジェクト「sample.txt」を削除します。コマンドを実行する前に、確認を求めるプロンプトが表示されます。-Force スイッチを追加すると、確認メッセージが表示されなくなります。

```
Remove-S3Object -BucketName test-files -Key sample.txt
```

例 2: このコマンドは、指定されたバージョンのオブジェクト「sample.txt」をバケット「test-files」から削除します。これは、バケットがオブジェクトバージョンを有効にするように設定されていることを前提としています。

```
Remove-S3Object -BucketName test-files -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

例 3: このコマンドは、バケット「test-files」からオブジェクト「sample1.txt」、「sample2.txt」、「sample3.txt」を 1 回のバッチ操作で削除します。このサービスの応答では、削除の成功またはエラーのステータスを問わず、処理されたすべてのキーがリスト表示されます。このサービスで処理できなかったキーのエラーのみを取得するには、-ReportErrorsOnly パラメータを追加します (このパラメータは -Quiet というエイリアスを使用して指定することもできます)。

```
Remove-S3Object -BucketName test-files -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

例 4: この例では、-KeyCollection パラメータを指定したインライン式を使用して、削除するオブジェクトのキーを取得します。Get-S3Object は Amazon.S3.Model.S3Object インスタンスのコレクションを返します。各インスタンスには、オブジェクトを識別する文字列型の Key メンバーがあります。

```
Remove-S3Object -bucketname "test-files" -KeyCollection (Get-S3Object "test-
files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

例 5: この例では、バケット内のキープレフィックス「prefix/subprefix」を持つすべてのオブジェクトを取得して削除します。受け取るオブジェクトは一度に 1 つずつ処理されることに注意が必要です。大規模なコレクションの場合は、コレクションをコマンドレットの -InputObject (エイリアスは -S3ObjectCollection) パラメータに渡して、サービスを 1 回呼び出すだけでバッチで削除できるようにすることを検討してください。

```
Get-S3Object -BucketName "test-files" -KeyPrefix "prefix/subprefix" | Remove-
S3Object -Force
```

例 6: この例では、削除マーカーを表す Amazon.S3.Model.S3ObjectVersion インスタンスのコレクションを、削除コマンドレットにパイプします。受け取るオブジェクトは一度に 1 つずつ処理されることに注意が必要です。大規模なコレクションの場合は、コレクションをコマンドレットの -InputObject (エイリアスは -S3ObjectCollection) パラメータに渡して、サービスを 1 回呼び出すだけでバッチで削除できるようにすることを検討してください。

```
(Get-S3Version -BucketName "test-files").Versions | Where {$_.IsDeleteMarker -eq
 "True"} | Remove-S3Object -Force
```

例 7: このスクリプトは、-KeyAndVersionCollection パラメータで使用するオブジェクトの配列を作成することで、オブジェクトセット (この場合は削除マーカー) をバッチで削除する方法を示しています。

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
 {$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
 $marker.VersionId } }
```

```
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeleteObjects」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクト キーのリストからオブジェクトのセットを削除します。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.

        :param bucket: The bucket that contains the objects. This is a Boto3
 Bucket
                       resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
```

```
                and any that could not be deleted.
        """
        try:
            response = bucket.delete_objects(
                Delete={"Objects": [{"Key": key} for key in object_keys]}
            )
            if "Deleted" in response:
                logger.info(
                    "Deleted objects '%s' from bucket '%s'.",
                    [del_obj["Key"] for del_obj in response["Deleted"]],
                    bucket.name,
                )
            if "Errors" in response:
                logger.warning(
                    "Could not delete objects '%s' from bucket '%s'.",
                    [
                        f"{del_obj['Key']}: {del_obj['Code']}"
                        for del_obj in response["Errors"]
                    ],
                    bucket.name,
                )
        except ClientError:
            logger.exception("Couldn't delete any objects from bucket %s.",
 bucket.name)
            raise
        else:
            return response
```

バケット内のすべてのオブジェクトを削除します。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key
```

```
    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.

        :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
        """
        try:
            bucket.objects.delete()
            logger.info("Emptied bucket '%s'.", bucket.name)
        except ClientError:
            logger.exception("Couldn't empty bucket '%s'.", bucket.name)
            raise
```

すべてのバージョンを削除することによって、バージョン管理されているオブジェクトを完全に削除します。

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「DeleteObjects」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
  answer = gets.chomp.downcase
  if answer == "y"
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「DeleteObjects」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
pub async fn delete_objects(client: &Client, bucket_name: &str) ->
 Result<Vec<String>, Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    let mut delete_objects: Vec<ObjectIdentifier> = vec![];
    for obj in objects.contents() {
        let obj_id = ObjectIdentifier::builder()
            .set_key(Some(obj.key().unwrap().to_string()))
            .build()
            .map_err(Error::from)?;
        delete_objects.push(obj_id);
    }

    let return_keys = delete_objects.iter().map(|o| o.key.clone()).collect();

    if !delete_objects.is_empty() {
        client
            .delete_objects()
            .bucket(bucket_name)
            .delete(
                Delete::builder()
                    .set_objects(Some(delete_objects))
                    .build()
                    .map_err(Error::from)?,
            )
            .send()
            .await?;
    }

    let objects: ListObjectsV2Output =
 client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(Error::unhandled(
            "There were still objects left in the bucket.",
        )),
    }
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「DeleteObjects」を参照してください。

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
public func deleteObjects(bucket: String, keys: [String]) async throws {
    let input = DeleteObjectsInput(
        bucket: bucket,
        delete: S3ClientTypes.Delete(
            objects: keys.map({ S3ClientTypes.ObjectIdentifier(key: $0) }),
            quiet: true
        )
    )

    do {
        let output = try await client.deleteObjects(input: input)

        // As of the last update to this example, any errors are returned
        // in the `output` object's `errors` property. If there are any
        // errors in this array, throw an exception. Once the error
        // handling is finalized in later updates to the AWS SDK for
        // Swift, this example will be updated to handle errors better.

        guard let errors = output.errors else {
            return  // No errors.
        }
```

```
            if errors.count != 0 {
                throw ServiceHandlerError.deleteObjectsError
            }
        } catch {
            throw error
        }
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「DeleteObjects」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeletePublicAccessBlock** を使用する

以下のコード例は、DeletePublicAccessBlock の使用方法を示しています。

CLI

AWS CLI

バケットのブロックパブリックアクセス設定を削除するには

次の delete-public-access-block の例では、指定したバケットのブロックパブリックアクセス設定を削除します。

```
aws s3api delete-public-access-block \
    --bucket my-bucket
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeletePublicAccessBlock」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定したバケットのブロックパブリックアクセス設定をオフにします。

```
Remove-S3PublicAccessBlock -BucketName 's3testbucket' -Force -Select
 '^BucketName'
```

出力:

```
s3testbucket
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeletePublicAccessBlock」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketAccelerateConfiguration** を使用する

以下のコード例は、GetBucketAccelerateConfiguration の使用方法を示しています。

CLI

AWS CLI

バケットの加速設定を取得するには

次の get-bucket-accelerate-configuration の例では、指定したバケットの加速設定を取得します。

```
aws s3api get-bucket-accelerate-configuration \
    --bucket my-bucket
```

出力:

```
{
    "Status": "Enabled"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetBucketAccelerateConfiguration」を参照してください。

PowerShell

Tools for PowerShell

例 1: 指定したバケットで転送加速設定が有効になっている場合、このコマンドは Enabled の値を返します。

```
Get-S3BucketAccelerateConfiguration -BucketName 's3testbucket'
```

出力:

```
Value
-----
Enabled
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketAccelerateConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketAcl** を使用する

以下のコード例は、GetBucketAcl の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- アクセスコントロールリスト (ACL) を管理する

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        /// <summary>
        /// Get the access control list (ACL) for the new bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to get the
        /// access control list (ACL) of the bucket.</param>
        /// <param name="newBucketName">The name of the newly created bucket.</
param>
        /// <returns>An S3AccessControlList.</returns>
        public static async Task<S3AccessControlList>
 GetACLForBucketAsync(IAmazonS3 client, string newBucketName)
        {
            // Retrieve bucket ACL to show that the ACL was properly applied to
            // the new bucket.
            GetACLResponse getACLResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = newBucketName,
            });

            return getACLResponse.AccessControlList;
        }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「GetBucketAcl」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
            s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << getGranteeTypeString(grantee.GetType()) <<
 std::endl;
            }
```

```
            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string.
*/

Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
```

```
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string.
*/

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                   "objects in this bucket, and read/write this "
                   "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }

    return "Permission unknown";
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「GetBucketAcl」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットのアクセス制御リストを取得します。

```
aws s3api get-bucket-acl --bucket my-bucket
```

出力:

```
{
    "Owner": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
            },
            "Permission": "FULL_CONTROL"
        }
    ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「GetBucketAcl」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <objectKey>

                Where:
                  bucketName - The Amazon S3 bucket to get the access control
 list (ACL) for.
                  objectKey - The object to get the ACL for.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
```

```
        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    public static String getBucketACL(S3Client s3, String objectKey, String
 bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
                System.out.format("  %s: %s\n", grant.grantee().id(),
 grant.permission());
                grantee = grant.grantee().id();
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「GetBucketAcl」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

ACL 許可を取得します。

```
import { GetBucketAclCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketAclCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「GetBucketAcl」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()
            logger.info(
                "Got ACL for bucket %s. Owner is %s.", self.bucket.name,
 acl.owner
            )
        except ClientError:
            logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
            raise
        else:
            return acl
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「GetBucketAcl」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketAnalyticsConfiguration** を使用する

以下のコード例は、GetBucketAnalyticsConfiguration の使用方法を示しています。

CLI

AWS CLI

特定の ID を持つバケットの分析設定を取得するには

次の get-bucket-analytics-configuration の例では、指定されたバケットと ID の分析設定を表示します。

```
aws s3api get-bucket-analytics-configuration \
    --bucket my-bucket \
    --id 1
```

出力:

```
{
    "AnalyticsConfiguration": {
        "StorageClassAnalysis": {},
        "Id": "1"
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetBucketAnalyticsConfiguration」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケット内の「testfilter」という名前の分析フィルターの詳細を返します。

```
Get-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId
 'testfilter'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketAnalyticsConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketCors** を使用する

以下のコード例は、GetBucketCors の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        /// <summary>
        /// Retrieve the CORS configuration applied to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to retrieve the CORS configuration.</param>
        /// <returns>The created CORS configuration object.</returns>
        private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
        {
```

```
            GetCORSConfigurationRequest request = new
 GetCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            var response = await client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
            return configuration;
        }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「GetBucketCors」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットの Cross-Origin Resource Sharing 設定を取得します。

```
aws s3api get-bucket-cors --bucket my-bucket
```

出力:

```
{
    "CORSRules": [
        {
            "AllowedHeaders": [
                "*"
            ],
            "ExposeHeaders": [
                "x-amz-server-side-encryption"
            ],
            "AllowedMethods": [
                "PUT",
                "POST",
                "DELETE"
            ],
            "MaxAgeSeconds": 3000,
```

```
            "AllowedOrigins": [
                "http://www.example.com"
            ]
        },
        {
            "AllowedHeaders": [
                "Authorization"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedMethods": [
                "GET"
            ],
            "AllowedOrigins": [
                "*"
            ]
        }
    ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「GetBucketCors」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットの CORS ポリシーを取得します。

```
import { GetBucketCorsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketCorsCommand({
    Bucket: "test-bucket",
  });
```

```
  try {
    const { CORSRules } = await client.send(command);
    CORSRules.forEach((cr, i) => {
      console.log(
        `\nCORSRule ${i + 1}`,
        `\n${"-".repeat(10)}`,
        `\nAllowedHeaders: ${cr.AllowedHeaders.join(" ")}`,
        `\nAllowedMethods: ${cr.AllowedMethods.join(" ")}`,
        `\nAllowedOrigins: ${cr.AllowedOrigins.join(" ")}`,
        `\nExposeHeaders: ${cr.ExposeHeaders.join(" ")}`,
        `\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`,
      );
    });
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「GetBucketCors」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
```

```
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_cors(self):
        """
        Get the CORS rules for the bucket.

        :return The CORS rules for the specified bucket.
        """
        try:
            cors = self.bucket.Cors()
            logger.info(
                "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
self.bucket.name
            )
        except ClientError:
            logger.exception(("Couldn't get CORS for bucket %s.",
self.bucket.name))
            raise
        else:
            return cors
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「GetBucketCors」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require "aws-sdk-s3"
```

```
# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
 configuration for the bucket.
  def get_cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
 why: #{e.message}"
    nil
  end

end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「GetBucketCors」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketEncryption** を使用する

以下のコード例は、GetBucketEncryption の使用方法を示しています。

CLI

AWS CLI

バケットのサーバー側の暗号化設定を取得するには

次の get-bucket-encryption の例では、バケット my-bucket のサーバー側の暗号化設定を取得します。

```
aws s3api get-bucket-encryption \
    --bucket my-bucket
```

出力:

```
{
    "ServerSideEncryptionConfiguration": {
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }
        ]
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetBucketEncryption」を参照してください。

PowerShell

Tools for PowerShell

例 1: のコマンドは、指定したバケットに関連付けられたすべてのサーバー側暗号化ルールを返します。

```
Get-S3BucketEncryption -BucketName 's3casetestbucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketEncryption」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketInventoryConfiguration** を使用する

以下のコード例は、GetBucketInventoryConfiguration の使用方法を示しています。

CLI

AWS CLI

バケットのインベントリ設定を取得するには

次の get-bucket-inventory-configuration の例では、ID 1 を持つ指定したバケットのインベントリ設定を取得します。

```
aws s3api get-bucket-inventory-configuration \
    --bucket my-bucket \
    --id 1
```

出力:

```
{
    "InventoryConfiguration": {
        "IsEnabled": true,
        "Destination": {
            "S3BucketDestination": {
                "Format": "ORC",
                "Bucket": "arn:aws:s3:::my-bucket",
                "AccountId": "123456789012"
            }
        },
        "IncludedObjectVersions": "Current",
        "Id": "1",
        "Schedule": {
            "Frequency": "Weekly"
        }
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetBucketInventoryConfiguration」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケット内の「testinventory」という名前のインベントリの詳細を返します。

```
Get-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId
 'testinventory'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketInventoryConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketLifecycleConfiguration** を使用する

以下のコード例は、GetBucketLifecycleConfiguration の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        /// <summary>
        /// Returns a configuration object for the supplied bucket name.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the GetLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">The name of the S3 bucket for which a
        /// configuration will be created.</param>
```

```
        /// <returns>Returns a new LifecycleConfiguration object.</returns>
        public static async Task<LifecycleConfiguration>
 RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)
        {
            var request = new GetLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「GetBucketLifecycleConfiguration」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットのライフサイクル設定を取得します。

```
aws s3api get-bucket-lifecycle-configuration --bucket my-bucket
```

出力:

```
{
    "Rules": [
        {
            "ID": "Move rotated logs to Glacier",
            "Prefix": "rotated/",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Date": "2015-11-10T00:00:00.000Z",
                    "StorageClass": "GLACIER"
                }
            ]
        },
        {
```

```
            "Status": "Enabled",
            "Prefix": "",
            "NoncurrentVersionTransitions": [
                {
                    "NoncurrentDays": 0,
                    "StorageClass": "GLACIER"
                }
            ],
            "ID": "Move old versions to Glacier"
        }
    ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「GetBucketLifecycleConfiguration」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_lifecycle_configuration(self):
        """
```

```
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()
            logger.info(
                "Got lifecycle rules %s for bucket '%s'.",
                config.rules,
                self.bucket.name,
            )
        except:
            logger.exception(
                "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
        else:
            return config.rules
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「GetBucketLicecycleConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketLocation** を使用する

以下のコード例は、GetBucketLocation の使用方法を示しています。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットの場所の制約を取得します (制約が存在する場合)。

```
aws s3api get-bucket-location --bucket my-bucket
```

出力:

```
{
    "LocationConstraint": "us-west-2"
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「GetBucketLocation」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、制約が存在する場合、バケット「s3testbucket」の場所の制約を返します。

```
Get-S3BucketLocation -BucketName 's3testbucket'
```

出力:

```
Value
-----
ap-south-1
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketLocation」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
async fn show_buckets(strict: bool, client: &Client, region: &str) -> Result<(),
 Error> {
    let resp = client.list_buckets().send().await?;
```

```
    let buckets = resp.buckets();
    let num_buckets = buckets.len();

    let mut in_region = 0;

    for bucket in buckets {
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint().unwrap().as_ref() == region {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「GetBucketLocation」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketLogging** を使用する

以下のコード例は、GetBucketLogging の使用方法を示しています。

CLI

AWS CLI

バケットのログ記録ステータスを取得するには

次の get-bucket-logging の例では、指定したバケットのログ記録ステータスを取得します。

```
aws s3api get-bucket-logging \
    --bucket my-bucket
```

出力:

```
{
    "LoggingEnabled": {
        "TargetPrefix": "",
        "TargetBucket": "my-bucket-logs"
          }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetBucketLogging」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定したバケットのログ記録ステータスを返します。

```
Get-S3BucketLogging -BucketName 's3testbucket'
```

出力:

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
```

```
testbucket1        {}     testprefix
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketLogging」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketMetricsConfiguration** を使用する

以下のコード例は、GetBucketMetricsConfiguration の使用方法を示しています。

CLI

AWS CLI

特定の ID を持つバケットのメトリクス設定を取得するには

次の get-bucket-metrics-configuration の例では、指定したバケットと ID のメトリクス設定を表示します。

```
aws s3api get-bucket-metrics-configuration \
    --bucket my-bucket \
    --id 123
```

出力:

```
{
    "MetricsConfiguration": {
        "Filter": {
            "Prefix": "logs"
        },
        "Id": "123"
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetBucketMetricsConfiguration」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットの「testfilter」という名前のメトリクスフィルターに関する詳細を返します。

```
Get-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId
 'testfilter'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketMetricsConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketNotification** を使用する

以下のコード例は、GetBucketNotification の使用方法を示しています。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットの通知設定を取得します。

```
aws s3api get-bucket-notification --bucket my-bucket
```

出力:

```
{
    "TopicConfiguration": {
        "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
        "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNl",
        "Event": "s3:ObjectCreated:*",
        "Events": [
            "s3:ObjectCreated:*"
        ]
    }
```

```
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetBucketNotification」を参照してください。

PowerShell

Tools for PowerShell

例 1: この例では、指定したバケットの通知設定を取得します。

```
Get-S3BucketNotification -BucketName kt-tools | select -ExpandProperty
 TopicConfigurations
```

出力:

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketNotification」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketPolicy** を使用する

以下のコード例は、GetBucketPolicy の使用方法を示しています。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                 const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
            s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n"
 <<
                  policy_stream.str() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ SDK for Rust API リファレンス」の「GetBucketPolicy」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットのバケットポリシーを取得します。

```
aws s3api get-bucket-policy --bucket my-bucket
```

出力:

```
{
    "Policy": "{\"Version\":\"2008-10-17\",\"Statement\":[{\"Sid\":\"\",\"Effect
\":\"Allow\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":
\"arn:aws:s3:::my-bucket/*\"},{\"Sid\":\"\",\"Effect\":\"Deny\",\"Principal\":
\"*\",\"Action\":\"s3:GetObject\",\"Resource\":\"arn:aws:s3:::my-bucket/secret/*
\"}]}"
}
```

次の例は、Amazon S3 バケットポリシーをダウンロードしてファイルを変更し、put-bucket-policy を使用して変更したバケットポリシーを適用する方法を示しています。バケットポリシーをファイルにダウンロードするには、以下を実行します。

```
aws s3api get-bucket-policy --bucket mybucket --query Policy --output text >
 policy.json
```

その後、必要に応じて policy.json ファイルを変更できます。最後に、次のコマンドを実行して、変更したポリシーを S3 バケットに適用することができます。

必要に応じて policy.json ファイル。最後に、次のコマンドを実行して、変更したポリシーを S3 バケットに適用することができます。

必要に応じてファイル。最後に、次のコマンドを実行して、変更したポリシーを S3 バケットに適用することができます。

```
aws s3api put-bucket-policy --bucket mybucket --policy file://policy.json
```

- API の詳細については、AWS CLI コマンドリファレンスの「GetBucketPolicy」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to get the policy from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }
```

```
    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x SDK for Rust API リファレンス」の「GetBucketPolicy」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットポリシーを取得します。

```
import { GetBucketPolicyCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});
```

```
export const main = async () => {
  const command = new GetBucketPolicyCommand({
    Bucket: "test-bucket",
  });

  try {
    const { Policy } = await client.send(command);
    console.log(JSON.parse(Policy));
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript SDK for Rust API リファレンス」の「GetBucketPolicy」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetBucketPolicy」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットに関連付けられたバケットポリシーを出力します。

```
Get-S3BucketPolicy -BucketName 's3testbucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketPolicy」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
    def get_policy(self):
        """
        Get the security policy of the bucket.

        :return: The security policy of the specified bucket, in JSON format.
        """
        try:
            policy = self.bucket.Policy()
            logger.info(
                "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get policy for bucket '%s'.",
 self.bucket.name)
            raise
        else:
            return json.loads(policy.policy)
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「GetBucketPolicy」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
```

```
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def get_policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    nil
  end

end
```

- API の詳細については、「AWS SDK for Ruby SDK for Rust API リファレンス」の「GetBucketPolicy」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketPolicyStatus** を使用する

以下のコード例は、GetBucketPolicyStatus の使用方法を示しています。

CLI

AWS CLI

バケットがパブリックかどうかを示すバケットのポリシーステータスを取得するには

次の get-bucket-policy-status の例では、バケット my-bucket のポリシーステータスを取得します。

```
aws s3api get-bucket-policy-status \
    --bucket my-bucket
```

出力:

```
{
    "PolicyStatus": {
        "IsPublic": false
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetBucketPolicyStatus」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットのポリシーステータスを返し、バケットがパブリックかどうかを示します。

```
Get-S3BucketPolicyStatus -BucketName 's3casetestbucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketPolicyStatus」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketReplication** を使用する

以下のコード例は、GetBucketReplication の使用方法を示しています。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットのレプリケーション設定を取得します。

```
aws s3api get-bucket-replication --bucket my-bucket
```

出力:

```
{
    "ReplicationConfiguration": {
        "Rules": [
            {
                "Status": "Enabled",
                "Prefix": "",
                "Destination": {
                    "Bucket": "arn:aws:s3:::my-bucket-backup",
                    "StorageClass": "STANDARD"
                },
                "ID": "ZmUwNzE4ZmQ4tMjVhOS00MTlkLOGI4NDkzZTIWJjNTUtYTA1"
            }
        ],
        "Role": "arn:aws:iam::123456789012:role/s3-replication-role"
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetBucketReplication」を参照してください。

PowerShell

Tools for PowerShell

例 1: mybucket」という名前のバケットに設定されているレプリケーション設定の情報を返します。

```
Get-S3BucketReplication -BucketName mybucket
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketReplication」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketRequestPayment** を使用する

以下のコード例は、GetBucketRequestPayment の使用方法を示しています。

CLI

AWS CLI

バケットのリクエストの支払い設定を取得するには

次の get-bucket-request-payment の例では、指定したバケットのリクエスタ支払いの設定を取得します。

```
aws s3api get-bucket-request-payment \
    --bucket my-bucket
```

出力:

```
{
    "Payer": "BucketOwner"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetBucketRequestPayment」を参照してください。

PowerShell

Tools for PowerShell

例 1:「mybucket」という名前のバケットのリクエストの支払い設定を返します。デフォルトでは、バケット所有者はバケットからのダウンロード料金を支払います。

```
Get-S3BucketRequestPayment -BucketName mybucket
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketRequestPayment」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketTagging** を使用する

以下のコード例は、GetBucketTagging の使用方法を示しています。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットのタグ付け設定を取得します。

```
aws s3api get-bucket-tagging --bucket my-bucket
```

出力:

```
{
    "TagSet": [
        {
            "Value": "marketing",
            "Key": "organization"
        }
    ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetBucketTagging」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定したバケットに関連付けられたすべてのタグを返します。

```
Get-S3BucketTagging -BucketName 's3casetestbucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketTagging」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketVersioning** を使用する

以下のコード例は、GetBucketVersioning の使用方法を示しています。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットのバージョニング設定を取得します。

```
aws s3api get-bucket-versioning --bucket my-bucket
```

出力:

```
{
    "Status": "Enabled"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetBucketVersioning」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定したバケットに関するバージョニングステータスを返します。

```
Get-S3BucketVersioning -BucketName 's3testbucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketVersioning」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketWebsite** を使用する

以下のコード例は、GetBucketWebsite の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
                // Get the website configuration.
                GetBucketWebsiteRequest getRequest = new
GetBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                };
                GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine($"Index document:
{getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
                Console.WriteLine($"Error document:
{getResponse.WebsiteConfiguration.ErrorDocument}");
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「GetBucketWebsite」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
```

```
                                    const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
            s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                  << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult =
 outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"
                  << std::endl
                  << "Index page : "
                  << websiteResult.GetIndexDocument().GetSuffix()
                  << std::endl
                  << "Error page: "
                  << websiteResult.GetErrorDocument().GetKey()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「GetBucketWebsite」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットの静的ウェブサイト設定を取得します。

```
aws s3api get-bucket-website --bucket my-bucket
```

出力:

```
{
    "IndexDocument": {
        "Suffix": "index.html"
    },
    "ErrorDocument": {
        "Key": "error.html"
    }
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「GetBucketWebsite」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

ウェブサイト設定を取得します。

```
import { GetBucketWebsiteCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketWebsiteCommand({
    Bucket: "test-bucket",
  });

  try {
    const { ErrorDocument, IndexDocument } = await client.send(command);
    console.log(
```

```
      `Your bucket is set up to host a website. It has an error document:`,
      `${ErrorDocument.Key}, and an index document: ${IndexDocument.Suffix}.`,
    );
  } catch (err) {
    console.error(err);
  }
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「GetBucketWebsite」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットの静的ウェブサイトの設定の詳細を返します。

```
Get-S3BucketWebsite -BucketName 's3testbucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetBucketWebsite」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetObject** を使用する

以下のコード例は、GetObject の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- バケットが変更されている場合、バケットからオブジェクトを取得する
- マルチリージョンアクセスポイント からオブジェクトを取得する
- バケットとオブジェクトの使用を開始する
- 暗号化の開始方法

- アップロードとダウンロードを追跡する

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        /// <summary>
        /// Shows how to download an object from an Amazon S3 bucket to the
        /// local computer.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket where the object is
        /// currently stored.</param>
        /// <param name="objectName">The name of the object to download.</param>
        /// <param name="filePath">The path, including filename, where the
        /// downloaded object will be stored.</param>
        /// <returns>A boolean value indicating the success or failure of the
        /// download process.</returns>
        public static async Task<bool> DownloadObjectFromBucketAsync(
            IAmazonS3 client,
            string bucketName,
            string objectName,
            string filePath)
        {
            // Create a GetObject request
            var request = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
            };

            // Issue request and remember to dispose of the response
            using GetObjectResponse response = await
client.GetObjectAsync(request);
```

```
            try
            {
                // Save object to local file
                await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
                return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error saving {objectName}: {ex.Message}");
                return false;
            }
        }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「GetObject」を参照してください。

Bash

Bash スクリプトを使用した AWS CLI

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
#############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function download_object_from_bucket
#
```

```
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetObject」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
            client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
                  << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「GetObject」を参照してください。

CLI

AWS CLI

次の例は get-object コマンドを使用して、Amazon S3 からオブジェクトをダウンロードします。

```
aws s3api get-object --bucket text-content --key dir/my_images.tar.bz2
 my_images.tar.bz2
```

outfile パラメータは、「--outfile」などのオプション名なしで指定されていることに注意してください。出力ファイルの名前はコマンドの最後のパラメータでなければなりません。

次の例は --range を使用して、オブジェクトから特定のバイト範囲をダウンロードする方法を示しています。バイト範囲には「bytes=」というプレフィックスを付ける必要があることに注意してください。

```
aws s3api get-object --bucket text-content --key dir/my_data --range
 bytes=8888-9999 my_data_range
```

オブジェクトの取得の詳細については、「Amazon S3 ユーザーガイド」の「オブジェクトを取得する」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetObject」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(bucketName string, objectKey string,
 fileName string) error {
 result, err := basics.S3Client.GetObject(context.TODO(), &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
```

```
    Key:    aws.String(objectKey),
   })
   if err != nil {
    log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
   objectKey, err)
    return err
   }
   defer result.Body.Close()
   file, err := os.Create(fileName)
   if err != nil {
    log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
    return err
   }
   defer file.Close()
   body, err := io.ReadAll(result.Body)
   if err != nil {
    log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
   err)
   }
   _, err = file.Write(body)
   return err
  }
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「GetObject」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

S3Client を使用して、データをバイト配列として読み取ります。

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <path>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                    path - The path where the file is written to.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
```

```
        getObjectBytes(s3, bucketName, keyName, path);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
 keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

S3TransferManager を使用して、ローカルファイルに S3 バケットのオブジェクトをダウンロードします。完全なファイルとテストを表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
```

```
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String
 bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
 transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
 downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
 downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

S3Client を使用して、オブジェクトに属するタグを読み取ります。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    public static void listTags(S3Client s3, String bucketName, String keyName) {
        try {
            GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();
```

```
            GetObjectTaggingResponse tags =
 s3.getObjectTagging(getTaggingRequest);
            List<Tag> tagSet = tags.tagSet();
            for (Tag tag : tagSet) {
                System.out.println(tag.key());
                System.out.println(tag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

S3Client を使用してオブジェクトの URL を取得します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.
```

```
                    keyName - A key name that represents the object.\s
                """";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for  " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

S3Client を使用する S3Presigner クライアントオブジェクトを使用してオブジェクトを取得します。

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
```

```
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents a text file.\s
                """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
                .region(region)
                .build();
```

```
        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(60))
                    .getObjectRequest(getObjectRequest)
                    .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
                values.forEach(value -> {
                    connection.addRequestProperty(header, value);
                });
            });

            // Send any request payload that the service needs (not needed when
            // isBrowserExecutable is true).
            if (presignedGetObjectRequest.signedPayload().isPresent()) {
                connection.setDoOutput(true);

                try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
                        OutputStream httpOutputStream =
connection.getOutputStream()) {
                    IoUtils.copy(signedPayload, httpOutputStream);
                }
            }
```

```
            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                System.out.println("Service returned response: ");
                IoUtils.copy(content, System.out);
            }

        } catch (S3Exception | IOException e) {
            e.getStackTrace();
        }
    }
}
```

ResponseTransformer オブジェクトと S3Client を使用してオブジェクトを取得します。

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetDataResponseTransformer {
    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <bucketName> <keyName> <path>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                    path - The path where the file is written to.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();
```

```
        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「GetObject」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトをダウンロードします。

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetObjectCommand({
    Bucket: "test-bucket",
    Key: "hello-s3.txt",
  });

  try {
    const response = await client.send(command);
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log(str);
```

```
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「GetObject」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetObject」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトを取得します。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「GetObject」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、バケット「test-files」から項目「sample.txt」を取得して、それを現在の場所にある「local-sample.txt」という名前のファイルに保存します。このコマンドを呼び出す前にファイル「local-sample.txt」が存在していなくても構いません。

```
Read-S3Object -BucketName test-files -Key sample.txt -File local-sample.txt
```

例 2: このコマンドは、バケット「test-files」から仮想ディレクトリ「DIR」を取得し、それを現在の場所の「Local-DIR」という名前のフォルダに保存します。このコマンドを呼び出す前に、フォルダー「Local-DIR」が存在していなくても構いません。

```
Read-S3Object -BucketName test-files -KeyPrefix DIR -Folder Local-DIR
```

例 3: バケット名に「config」を含むバケットから、キーが「.json」で終わるすべてのオブジェクトを、指定したフォルダ内のファイルにダウンロードします。オブジェクトキーはファイル名の設定に使用されます。

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetObject」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class ObjectWrapper:
```

```
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get(self):
        """
        Gets the object.

        :return: The object data in bytes.
        """
        try:
            body = self.object.get()["Body"].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        else:
            return body
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「GetObject」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトを取得します。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is
 downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
```

```
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes} downloaded to
 #{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

オブジェクトを取得し、サーバー側の暗号化状態をレポートします。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data
```

```
  encryption = obj_data.server_side_encryption.nil? ? "no" :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「GetObject」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
async fn get_object(client: Client, opt: Opt) -> Result<usize, anyhow::Error> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone())?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
        .send()
        .await?;

    let mut byte_count = 0_usize;
    while let Some(bytes) = object.body.try_next().await? {
        let bytes_len = bytes.len();
        file.write_all(&bytes)?;
        trace!("Intermediate write of {bytes_len}");
```

```
        byte_count += bytes_len;
    }

    Ok(byte_count)
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「GetObject」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        oo_result = lo_s3->getobject(           " oo_result is returned for
testing purposes. "
                    iv_bucket = iv_bucket_name
                    iv_key = iv_object_key
                ).
        DATA(lv_object_data) = oo_result->get_body( ).
        MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「GetObject」を参照してください。

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットからローカルファイルにオブジェクトをダウンロードします。

```
    public func downloadFile(bucket: String, key: String, to: String) async
throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the data stream object. Return immediately if there isn't one.
        guard let body = output.body,
              let data = try await body.readData() else {
            return
        }
        try data.write(to: fileUrl)
    }
```

オブジェクトを Swift Data オブジェクトに読み込みます。

```
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
```

```
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the stream and return its contents in a `Data` object. If
        // there is no stream, return an empty `Data` object instead.
        guard let body = output.body,
              let data = try await body.readData() else {
            return "".data(using: .utf8)!
        }

        return data
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「GetObject」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で `GetObjectAcl` を使用する

以下のコード例は、`GetObjectAcl` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- アクセスコントロールリスト (ACL) を管理する

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
            s3Client.GetObjectAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObjectAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            std::cout << "For object " << objectKey << ": "
                      << std::endl << std::endl;

            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << getGranteeTypeString(grantee.GetType()) <<
 std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
```

```
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
*/
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
```

```
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string
*/
Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can read this object's data and its metadata, "
                   "and read/write this object's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can read this object's data and its metadata";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this object's permissions";
            // case Aws::S3::Model::Permission::WRITE // Not applicable.
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this object's permissions";
        default:
            return "Permission unknown";
    }
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「GetObjectAcl」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケット内のオブジェクトのアクセス制御リストを取得します。

```
aws s3api get-object-acl --bucket my-bucket --key index.html
```

出力:

```
{
    "Owner": {
        "DisplayName": "my-username",
```

```
            "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
        },
        "Grants": [
            {
                "Grantee": {
                    "DisplayName": "my-username",
                    "ID":
 "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
                },
                "Permission": "FULL_CONTROL"
            },
            {
                "Grantee": {
                    "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
                },
                "Permission": "READ"
            }
        ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「GetObjectAcl」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
```

```
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetObjectAcl」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get_acl(self):
        """
        Gets the ACL of the object.
```

```
        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
            logger.info(
                "Got ACL for object %s owned by %s.",
                self.object.key,
                acl.owner["DisplayName"],
            )
        except ClientError:
            logger.exception("Couldn't get ACL for object %s.", self.object.key)
            raise
        else:
            return acl
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「GetObjectAAcl」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetObjectLegalHold** を使用する

以下のコード例は、GetObjectLegalHold の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- オブジェクトのリーガルホールド設定を取得する
- Amazon S3 オブジェクトをロックする

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
{bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }
```

- API の詳細については、AWS SDK for .NET API リファレンスの「GetObjectLegalHold」を参照してください。

CLI

AWS CLI

オブジェクトのリーガルホールドステータスを取得する

次の get-object-legal-hold の例では、指定したオブジェクトのリーガルホールドステータスを取得します。

```
aws s3api get-object-legal-hold \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf
```

出力:

```
{
    "LegalHold": {
        "Status": "ON"
    }
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「GetObjectLegalHold」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
```

```
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「GetObjectLegalHold」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   // Get the legal hold details for an S3 object.
   public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
       try {
           GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
               .bucket(bucketName)
               .key(objectKey)
               .build();

           GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
           System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
               ":\n\tStatus: " + response.legalHold().status());
           return response.legalHold();

       } catch (S3Exception ex) {
           System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
       }

       return null;
   }
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「GetObjectLegalHold」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetObjectLockConfiguration** を使用する

以下のコード例は、GetObjectLockConfiguration の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Amazon S3 オブジェクトをロックする

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
    public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
    {
        try
        {
            var request = new GetObjectLockConfigurationRequest()
            {
                BucketName = bucketName
            };

            var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
```

```
            Console.WriteLine($"\tBucket object lock config for {bucketName} in
 {bucketName}: " +
                              $"\n\tEnabled:
 {response.ObjectLockConfiguration.ObjectLockEnabled}" +
                              $"\n\tRule:
 {response.ObjectLockConfiguration.Rule?.DefaultRetention}");

            return response.ObjectLockConfiguration;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock config:
 '{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「GetObjectLockConfiguration」を参照してください。

CLI

AWS CLI

バケットのオブジェクトロック設定を取得するには

次の get-object-lock-configuration 例では、指定されたバケットのオブジェクトロック設定を取得します。

```
aws s3api get-object-lock-configuration \
    --bucket my-bucket-with-object-lock
```

出力:

```
{
    "ObjectLockConfiguration": {
        "ObjectLockEnabled": "Enabled",
        "Rule": {
            "DefaultRetention": {
                "Mode": "COMPLIANCE",
                "Days": 50
```

```
            }
        }
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetObjectLockConfiguration」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }

 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
```

```
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
  "ObjectLockConfigurationNotFoundError" {
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  } else {
   lockConfig = output.ObjectLockConfiguration
  }

  return lockConfig, err
 }
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「GetObjectLockConfiguration」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
 GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
 getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
 "+response.objectLockConfiguration().objectLockEnabled());
```

```
        System.out.println("\tRule: "+
 response.objectLockConfiguration().rule().defaultRetention());
    }
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「GetObjectLockConfiguration」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import {
  GetObjectLockConfigurationCommand,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 */
export const main = async (client, bucketName) => {
  const command = new GetObjectLockConfigurationCommand({
    Bucket: bucketName,
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
  });

  try {
    const { ObjectLockConfiguration } = await client.send(command);
    console.log(`Object Lock Configuration: ${ObjectLockConfiguration}`);
  } catch (err) {
```

```
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME");
}
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「GetObjectLockConfiguration」を参照してください。

PowerShell

Tools for PowerShell

例 1: 指定した S3 バケットのオブジェクトロック設定が有効になっている場合、このコマンドは値「Enabled」を返しま。

```
Get-S3ObjectLockConfiguration -BucketName 's3buckettesting' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

出力:

```
Value
-----
Enabled
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetObjectLockConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetObjectRetention** を使用する

以下のコード例は、GetObjectRetention の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Amazon S3 オブジェクトをロックする

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in
{bucketName}: " +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
```

```
            Console.WriteLine($"\tUnable to fetch object lock retention:
 '{ex.Message}'");
            return new ObjectLockRetention();
        }
    }
```

- API の詳細については、AWS SDK for .NET API リファレンスの「GetObjectRetention」を参照してください。

CLI

AWS CLI

オブジェクトのオブジェクト保持設定を取得するには

次の get-object-retention 例では、指定されたオブジェクトの保持設定を取得します。

```
aws s3api get-object-retention \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf
```

出力:

```
{
    "Retention": {
        "Mode": "GOVERNANCE",
        "RetainUntilDate": "2025-01-01T00:00:00.000Z"
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetObjectRetention」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectRetention retrieves the object retention configuration for an S3
 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
```

```
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「GetObjectRetention」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   // Get the retention period for an S3 object.
   public ObjectLockRetention getObjectRetention(String bucketName, String key){
       try {
           GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

           GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
           System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
           return response.retention();
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「GetObjectRetention」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import { GetObjectRetentionCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new GetObjectRetentionCommand({
    Bucket: bucketName,
    Key: objectKey,
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
```

```
    const { Retention } = await client.send(command);
    console.log(`Object Retention Settings: ${Retention.Status}`);
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME", "OBJECT_KEY");
}
```

- API の詳細については、AWS SDK for JavaScript API リファレンスの「GetObjectRetention」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、オブジェクトが保持されるまでのモードと日付を返します。

```
Get-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetObjectRetention」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetObjectTagging** を使用する

以下のコード例は、GetObjectTagging の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- タグの使用開始

CLI

AWS CLI

オブジェクトにアタッチされたタグを取得するには

次の get-object-tagging 指定したオブジェクトから指定したキーの値を取得します。

```
aws s3api get-object-tagging \
    --bucket my-bucket \
    --key doc1.rtf
```

出力:

```
{
    "TagSet": [
        {
            "Value": "confidential",
            "Key": "designation"
        }
    ]
}
```

次の get-object-tagging の例では、タグのないオブジェクト doc2.rtf のタグセットの取得を試行します。

```
aws s3api get-object-tagging \
    --bucket my-bucket \
    --key doc2.rtf
```

出力:

```
{
    "TagSet": []
}
```

次の get-object-tagging の例では、複数のタグがあるオブジェクト doc3.rtf のタグセットの取得を試行します。

```
aws s3api get-object-tagging \
    --bucket my-bucket \
```

```
    --key doc3.rtf
```

出力:

```
{
    "TagSet": [
        {
            "Value": "confidential",
            "Key": "designation"
        },
        {
            "Value": "finance",
            "Key": "department"
        },
        {
            "Value": "payroll",
            "Key": "team"
        }
    ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetObjectTagging」を参照してください。

PowerShell

Tools for PowerShell

例 1: この例では、指定した S3 バケット上に存在するオブジェクトに関連付けられたタグを返します。

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'testbucket123'
```

出力:

```
Key  Value
---  -----
test value
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetObjectTagging」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetPublicAccessBlock** を使用する

以下のコード例は、GetPublicAccessBlock の使用方法を示しています。

CLI

AWS CLI

バケットのブロックパブリックアクセス設定を指定または削除するには

次の get-public-access-block の例では、指定したバケットのブロックパブリックアクセス設定を表示します。

```
aws s3api get-public-access-block \
    --bucket my-bucket
```

出力:

```
{
    "PublicAccessBlockConfiguration": {
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "BlockPublicAcls": true,
        "RestrictPublicBuckets": true
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetPublicAccessBlock」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットのパブリックアクセスブロック設定を返します。

```
Get-S3PublicAccessBlock -BucketName 's3testbucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「GetPublicAccessBlock」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **HeadBucket** を使用する

以下のコード例は、HeadBucket の使用方法を示しています。

Bash

Bash スクリプトを使用した AWS CLI

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
###############################################################################
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#       $1 - The name of the bucket to check.
#
# Returns:
#       0 - If the bucket already exists.
#       1 - If the bucket doesn't exist.
###############################################################################
function bucket_exists() {
  local bucket_name
  bucket_name=$1

  # Check whether the bucket already exists.
```

```
  # We suppress all output - we're interested only in the return code.

  if aws s3api head-bucket \
    --bucket "$bucket_name" \
    >/dev/null 2>&1; then
    return 0 # 0 in Bash script means true.
  else
    return 1 # 1 in Bash script means false.
  fi
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「HeadBucket」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットへのアクセスを確認します。

```
aws s3api head-bucket --bucket my-bucket
```

バケットが存在し、そのバケットにアクセスできる場合、出力は返されません。それ以外の場合は、エラーメッセージが表示されます。例:

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「HeadBucket」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(bucketName string) (bool, error) {
 _, err := basics.S3Client.HeadBucket(context.TODO(), &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
    log.Printf("Either you don't have access to bucket %v or another error
 occurred. "+
     "Here's what happened: %v\n", bucketName, err)
   }
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「HeadBucket」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def exists(self):
        """
        Determine whether the bucket exists and you have access to it.

        :return: True when the bucket exists; otherwise, False.
        """
        try:
            self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
            logger.info("Bucket %s exists.", self.bucket.name)
            exists = True
        except ClientError:
            logger.warning(
                "Bucket %s doesn't exist or you don't have access to it.",
                self.bucket.name,
            )
```

```
            exists = False
        return exists
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「HeadBucket」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **HeadObject** を使用する

以下のコード例は、HeadObject の使用方法を示しています。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケット内のオブジェクトのメタデータを取得します。

```
aws s3api head-object --bucket my-bucket --key index.html
```

出力:

```
{
    "AcceptRanges": "bytes",
    "ContentType": "text/html",
    "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
    "ContentLength": 77,
    "VersionId": "null",
    "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
    "Metadata": {}
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「HeadObject」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトのコンテンツタイプを調べる

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    public static void getContentType(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

オブジェクトの復元ステータスを取得する

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    public static void checkStatus(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
 response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「HeadObject」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object
 #{@object.bucket.name}:#{@object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
```

```
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「HeadObject」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ListBucketAnalyticsConfigurations** を使用する

以下のコード例は、ListBucketAnalyticsConfigurations の使用方法を示しています。

CLI

AWS CLI

バケットの分析設定のリストを取得するには

次の list-bucket-analytics-configurations の例では、指定したバケットの分析設定のリストを取得します。

```
aws s3api list-bucket-analytics-configurations \
    --bucket my-bucket
```

出力:

```
{
    "AnalyticsConfigurationList": [
        {
```

```
            "StorageClassAnalysis": {},
            "Id": "1"
        }
    ],
    "IsTruncated": false
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「ListBucketAnalyticsConfigurations」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットの最初の 100 つの分析設定を返します。

```
Get-S3BucketAnalyticsConfigurationList -BucketName 's3casetestbucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「ListBucketAnalyticsConfigurations」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ListBucketInventoryConfigurations** を使用する

以下のコード例は、ListBucketInventoryConfigurations の使用方法を示しています。

CLI

AWS CLI

バケットのインベントリ設定のリストを取得するには

次の list-bucket-inventory-configurations の例では、指定したバケットのインベントリ設定をリスト表示します。

```
aws s3api list-bucket-inventory-configurations \
```

```
    --bucket my-bucket
```

出力:

```
{
    "InventoryConfigurationList": [
        {
            "IsEnabled": true,
            "Destination": {
                "S3BucketDestination": {
                    "Format": "ORC",
                    "Bucket": "arn:aws:s3:::my-bucket",
                    "AccountId": "123456789012"
                }
            },
            "IncludedObjectVersions": "Current",
            "Id": "1",
            "Schedule": {
                "Frequency": "Weekly"
            }
        },
        {
            "IsEnabled": true,
            "Destination": {
                "S3BucketDestination": {
                    "Format": "CSV",
                    "Bucket": "arn:aws:s3:::my-bucket",
                    "AccountId": "123456789012"
                }
            },
            "IncludedObjectVersions": "Current",
            "Id": "2",
            "Schedule": {
                "Frequency": "Daily"
            }
        }
    ],
    "IsTruncated": false
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「ListBucketInventoryConfigurations」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットの最初の 100 つのインベントリ設定を返します。

```
Get-S3BucketInventoryConfigurationList -BucketName 's3testbucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「ListBucketInventoryConfigurations」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で `ListBuckets` を使用する

以下のコード例は、`ListBuckets` の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
    /// Service (Amazon S3) buckets belonging to the default account.
```

```
    /// </summary>
    public class ListBuckets
    {
        private static IAmazonS3 _s3Client;

        /// <summary>
        /// Get a list of the buckets owned by the default user.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <returns>The response from the ListingBuckets call that contains a
        /// list of the buckets owned by the default user.</returns>
        public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3
 client)
        {
            return await client.ListBucketsAsync();
        }

        /// <summary>
        /// This method lists the name and creation date for the buckets in
        /// the passed List of S3 buckets.
        /// </summary>
        /// <param name="bucketList">A List of S3 bucket objects.</param>
        public static void DisplayBucketList(List<S3Bucket> bucketList)
        {
            bucketList
                .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
 created on: {b.CreationDate}"));
        }

        public static async Task Main()
        {
            // The client uses the AWS Region of the default user.
            // If the Region where the buckets were created is different,
            // pass the Region to the client constructor. For example:
            // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
            _s3Client = new AmazonS3Client();
            var response = await GetBuckets(_s3Client);
            DisplayBucketList(response.Buckets);
        }
    }
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「ListBuckets」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
 buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「ListBuckets」を参照してください。

CLI

AWS CLI

次のコマンドは、list-buckets コマンドを使用してすべての Amazon S3 バケット (すべてのリージョン) の名前を表示します。

```
aws s3api list-buckets --query "Buckets[].Name"
```

クエリオプションで list-buckets の出力をバケット名のみにフィルタリングします。

バケットの詳細については、「Amazon S3 ユーザーガイド」の「バケットの使用」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「ListBuckets」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
```

```
func (basics BucketBasics) ListBuckets() ([]types.Bucket, error) {
 result, err := basics.S3Client.ListBuckets(context.TODO(),
 &s3.ListBucketsInput{})
 var buckets []types.Bucket
 if err != nil {
  log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
 } else {
  buckets = result.Buckets
 }
 return buckets, err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「ListBuckets」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);

    }
    public static void listAllBuckets(S3Client s3) {
        ListBucketsResponse response = s3.listBuckets();
        List<Bucket> bucketList = response.buckets();
        for (Bucket bucket: bucketList) {
            System.out.println("Bucket name "+bucket.name());
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「ListBuckets」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットを一覧表示します。

```
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
```

```
  const command = new ListBucketsCommand({});

  try {
    const { Owner, Buckets } = await client.send(command);
    console.log(
      `${Owner.DisplayName} owns ${Buckets.length} bucket${
        Buckets.length === 1 ? "" : "s"
      }:`,
    );
    console.log(`${Buckets.map((b) => ` • ${b.Name}`).join("\n")}`);
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「ListBuckets」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドはすべての S3 バケットを返します。

```
Get-S3Bucket
```

例 2: このコマンドは「test-files」という名前のバケットを返します。

```
Get-S3Bucket -BucketName test-files
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「ListBuckets」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    @staticmethod
    def list(s3_resource):
        """
        Get the buckets in all Regions for the current account.

        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
 Boto3
                            that contains collections and factory methods to
 create
                            other high-level S3 sub-resources.
        :return: The list of buckets.
        """
        try:
            buckets = list(s3_resource.buckets.all())
            logger.info("Got buckets: %s.", buckets)
        except ClientError:
            logger.exception("Couldn't get buckets.")
            raise
        else:
```

```
            return buckets
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「ListBuckets」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require "aws-sdk-s3"

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts "Found these buckets:"
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
```

```
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「ListBuckets」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
async fn show_buckets(strict: bool, client: &Client, region: &str) -> Result<(),
 Error> {
    let resp = client.list_buckets().send().await?;
    let buckets = resp.buckets();
    let num_buckets = buckets.len();

    let mut in_region = 0;

    for bucket in buckets {
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;
```

```
            if r.location_constraint().unwrap().as_ref() == region {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「ListBuckets」を参照してください。

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
/// Return an array containing information about every available bucket.
///
/// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing
///   each bucket.
public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {
    let output = try await client.listBuckets(input: ListBucketsInput())

    guard let buckets = output.buckets else {
        return []
    }
    return buckets
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「ListTables」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ListMultipartUploads** を使用する

以下のコード例は、ListMultipartUploads の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 不完全なマルチパートアップロードを特定する

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットのすべてのアクティブなマルチパートアップロードを一覧表示します。

```
aws s3api list-multipart-uploads --bucket my-bucket
```

出力:

```
{
    "Uploads": [
        {
            "Initiator": {
                "DisplayName": "username",
                "ID": "arn:aws:iam::0123456789012:user/username"
            },
            "Initiated": "2015-06-02T18:01:30.000Z",
            "UploadId":
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
            "StorageClass": "STANDARD",
            "Key": "multipart/01",
            "Owner": {
                "DisplayName": "aws-account-name",
                "ID":
 "100719349fc3b6dcd7c820a124bf7aecd408092c3d7b51b38494939801fc248b"
            }
        }
    ],
    "CommonPrefixes": []
}
```

進行中のマルチパートアップロードでは、Amazon S3 のストレージコストが発生します。アクティブなマルチパートアップロードを完了または中止して、その一部をアカウントから削除します。

- API の詳細については、AWS CLI コマンドリファレンスの「ListMultipartUploads」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key()
 + "\", id = " + upload.uploadId());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「ListMultipartUploads」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ListObjectVersions** を使用する

以下のコード例は、ListObjectVersions の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- バージョン管理されたオブジェクトを操作する

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example lists the versions of the objects in a version enabled
    /// Amazon Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectVersions
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";

            // If the AWS Region where your bucket is defined is different from
            // the AWS Region where the Amazon S3 bucket is defined, pass the
constant
            // for the AWS Region to the client constructor like this:
            //      var client = new AmazonS3Client(RegionEndpoint.USWest2);
            IAmazonS3 client = new AmazonS3Client();
            await GetObjectListWithAllVersionsAsync(client, bucketName);
```

```
        }

        /// <summary>
        /// This method lists all versions of the objects within an Amazon S3
        /// version enabled bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// ListVersionsAsync.</param>
        /// <param name="bucketName">The name of the version enabled Amazon S3
 bucket
        /// for which you want to list the versions of the contained objects.</
param>
        public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
 client, string bucketName)
        {
            try
            {
                // When you instantiate the ListVersionRequest, you can
                // optionally specify a key name prefix in the request
                // if you want a list of object versions of a specific object.

                // For this example we set a small limit in MaxKeys to return
                // a small list of versions.
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    MaxKeys = 2,
                };

                do
                {
                    ListVersionsResponse response = await
 client.ListVersionsAsync(request);

                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine($"key: {entry.Key} size:
 {entry.Size}");
                    }

                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
```

```
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
                    else
                    {
                        request = null;
                    }
                }
                while (request != null);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}'");
            }
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「ListObjectVersions」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケット内のオブジェクトのバージョン情報を取得します。

```
aws s3api list-object-versions --bucket my-bucket --prefix index.html
```

出力:

```
{
    "DeleteMarkers": [
        {
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
```

```
            "IsLatest": true,
            "VersionId": "B2VsEK5saUNNHKcOAJj7hIE86RozToyq",
            "Key": "index.html",
            "LastModified": "2015-11-10T00:57:03.000Z"
        },
        {
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
            "Key": "index.html",
            "LastModified": "2015-11-09T23:32:20.000Z"
        }
    ],
    "Versions": [
        {
            "LastModified": "2015-11-10T00:20:11.000Z",
            "VersionId": "Rb_l2T8UHDkFEwCgJjhlgPOZC0qJ.vpD",
            "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 38
        },
        {
            "LastModified": "2015-11-09T23:26:41.000Z",
            "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAFf",
            "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
```

```
            "Size": 38
        },
        {
            "LastModified": "2015-11-09T22:50:50.000Z",
            "VersionId": "null",
            "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 533823
        }
    ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「ListObjectVersions」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}
```

```
// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「ListObjectVersions」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;
```

```
    for version in resp.versions() {
        println!("{}", version.key().unwrap_or_default());
        println!("  version ID: {}", version.version_id().unwrap_or_default());
        println!();
    }

    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「ListObjectVersions」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ListObjects** を使用する

以下のコード例は、ListObjects の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Amazon S3 オブジェクトを一覧表示するウェブページを作成する

CLI

AWS CLI

次の例は、list-objects コマンドを使用して、指定されたバケット内のすべてのオブジェクトの名前を表示します。

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size:
 Size}'
```

この例では、--query 引数を使用して list-objects の出力を各オブジェクトのキー値とサイズでフィルタリングしています。

オブジェクトの詳細については、「Amazon S3 デベロッパーガイド」の「Working with Amazon S3 Objects」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「ListObjects」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、バケット「test-files」内のすべての項目に関する情報を取得します。

```
Get-S3Object -BucketName test-files
```

例 2: このコマンドは、バケット「test-files」内の項目「sample.txt」に関する情報を取得します。

```
Get-S3Object -BucketName test-files -Key sample.txt
```

例 3: このコマンドは、バケット「test-files」からプレフィックス「sample」を持つすべての項目に関する情報を取得します。

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンスの「ListObjects」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で `ListObjectsV2` を使用する

以下のコード例は、`ListObjectsV2` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- バケットとオブジェクトの使用を開始する

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        /// <summary>
        /// Shows how to list the objects in an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket for which to list
        /// the contents.</param>
        /// <returns>A boolean value indicating the success or failure of the
        /// copy operation.</returns>
        public static async Task<bool> ListBucketContentsAsync(IAmazonS3 client,
string bucketName)
        {
            try
            {
                var request = new ListObjectsV2Request
                {
                    BucketName = bucketName,
                    MaxKeys = 5,
                };

                Console.WriteLine("--------------------------------------");
                Console.WriteLine($"Listing the contents of {bucketName}:");
                Console.WriteLine("--------------------------------------");

                ListObjectsV2Response response;

                do
                {
                    response = await client.ListObjectsV2Async(request);

                    response.S3Objects
```

```
                        .ForEach(obj => Console.WriteLine($"{obj.Key,-35}
{obj.LastModified.ToShortDateString(),10}{obj.Size,10}"));

                    // If the response is truncated, set the request
 ContinuationToken
                    // from the NextContinuationToken property of the response.
                    request.ContinuationToken = response.NextContinuationToken;
                }
                while (response.IsTruncated);

                return true;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error encountered on server.
 Message:'{ex.Message}' getting list of objects.");
                return false;
            }
        }
```

ページネーターを使用してオブジェクトを一覧表示します。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// The following example lists objects in an Amazon Simple Storage
    /// Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectsPaginator
    {
        private const string BucketName = "doc-example-bucket";

        public static async Task Main()
        {
            IAmazonS3 s3Client = new AmazonS3Client();

            Console.WriteLine($"Listing the objects contained in {BucketName}:
\n");
```

```
            await ListingObjectsAsync(s3Client, BucketName);
        }

        /// <summary>
        /// This method uses a paginator to retrieve the list of objects in an
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the S3 bucket whose objects
        /// you want to list.</param>
        public static async Task ListingObjectsAsync(IAmazonS3 client, string
bucketName)
        {
            var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
                BucketName = bucketName,
            });

            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
                Console.WriteLine($"Number of Keys: {response.KeyCount}");
                foreach (var entry in response.S3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「ListObjectsV2」を参照してください。

Bash

Bash スクリプトを使用した AWS CLI

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
#############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
#############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
```

```
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}
```

- API の詳細については、「AWS CLI API リファレンス」の「ListObjectsV2」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }
```

```
        auto outcome = s3Client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                    outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " object(s) found:" << std::endl;

    for (const auto &object: allObjects) {
        std::cout << "  " << object.GetKey() << std::endl;
    }

    return true;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「ListObjectsV2」を参照してください。

CLI

AWS CLI

バケット内のオブジェクトのリストを取得するには

次の `list-objects-v2` の例では、指定したバケットのオブジェクトを一覧表示します。

```
aws s3api list-objects-v2 \
    --bucket my-bucket
```

出力:

```
{
```

```
    "Contents": [
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"621503c373607d548b37cff8778d992c\"",
            "StorageClass": "STANDARD",
            "Key": "doc1.rtf",
            "Size": 391
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"a2cecc36ab7c7fe3a71a273b9d45b1b5\"",
            "StorageClass": "STANDARD",
            "Key": "doc2.rtf",
            "Size": 373
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"08210852f65a2e9cb999972539a64d68\"",
            "StorageClass": "STANDARD",
            "Key": "doc3.rtf",
            "Size": 399
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"d1852dd683f404306569471af106988e\"",
            "StorageClass": "STANDARD",
            "Key": "doc4.rtf",
            "Size": 6225
        }
    ]
}
```

- API の詳細については、「AWS CLI API リファレンス」の「ListObjectsV2」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(bucketName string) ([]types.Object, error)
 {
 result, err := basics.S3Client.ListObjectsV2(context.TODO(),
 &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 })
 var contents []types.Object
 if err != nil {
  log.Printf("Couldn't list objects in bucket %v. Here's why: %v\n", bucketName,
 err)
 } else {
  contents = result.Contents
 }
 return contents, err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「ListObjectsV2」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class ListObjects {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The Amazon S3 bucket from which objects are
 read.\s
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                    .builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                System.out.print("\n The object is " + calKb(myValue.size()) + "
 KBs");
                System.out.print("\n The owner is " + myValue.owner());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // convert bytes to kbs.
    private static long calKb(Long val) {
        return val / 1024;
    }
}
```

ページネーションを使用してオブジェクトを一覧表示します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The Amazon S3 bucket from which objects are
 read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                    .bucket(bucketName)
                    .maxKeys(1)
                    .build();
```

```
            ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
            listRes.stream()
                    .flatMap(r -> r.contents().stream())
                    .forEach(content -> System.out.println(" Key: " +
 content.key() + " size = " + content.size()));

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「ListObjectsV2」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケット内のすべてのオブジェクトを一覧表示します。オブジェクトが複数ある場合は、IsTruncated と NextContinuationToken を使用してリスト全体を繰り返し処理します。

```
import {
  S3Client,
  // This command supersedes the ListObjectsCommand and is the recommended way to
 list objects.
  ListObjectsV2Command,
} from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
```

```
  const command = new ListObjectsV2Command({
    Bucket: "my-bucket",
    // The default and maximum number of keys returned is 1000. This limits it to
    // one for demonstration purposes.
    MaxKeys: 1,
  });

  try {
    let isTruncated = true;

    console.log("Your bucket contains the following objects:\n");
    let contents = "";

    while (isTruncated) {
      const { Contents, IsTruncated, NextContinuationToken } =
        await client.send(command);
      const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
      contents += contentsList + "\n";
      isTruncated = IsTruncated;
      command.input.ContinuationToken = NextContinuationToken;
    }
    console.log(contents);
  } catch (err) {
    console.error(err);
  }
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「ListObjectsV2」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListObjectsV2」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケット内のオブジェクトを一覧表示します。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
```

```
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「ListObjectsV2」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.
```

```
        :param bucket: The bucket to query. This is a Boto3 Bucket resource.
        :param prefix: When specified, only objects that start with this prefix
 are listed.
        :return: The list of objects.
        """
        try:
            if not prefix:
                objects = list(bucket.objects.all())
            else:
                objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info(
                "Got objects %s from bucket '%s'", [o.key for o in objects],
 bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get objects for bucket '%s'.",
 bucket.name)
            raise
        else:
            return objects
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「ListObjectsV2」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket
```

```
  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why:
 #{e.message}"
    0
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「ListObjectsV2」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
pub async fn list_objects(client: &Client, bucket: &str) -> Result<(), Error> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「ListObjectsV2」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        oo_result = lo_s3->listobjectsv2(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name
        ).
        MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「ListObjectsV2」を参照してください。

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public func listBucketFiles(bucket: String) async throws -> [String] {
        let input = ListObjectsV2Input(
            bucket: bucket
        )
        let output = try await client.listObjectsV2(input: input)
        var names: [String] = []

        guard let objList = output.contents else {
            return []
        }

        for obj in objList {
            if let objName = obj.key {
                names.append(objName)
            }
        }

        return names
    }
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「ListObjectsV2」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketAccelerateConfiguration** を使用する

以下のコード例は、PutBucketAccelerateConfiguration の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
/// bucket-level feature that enables you to perform faster data transfers
/// to Amazon S3. This example shows how to configure Transfer
/// Acceleration.
/// </summary>
public class TransferAcceleration
{
    /// <summary>
    /// The main method initializes the client object and sets the
    /// Amazon Simple Storage Service (Amazon S3) bucket name before
    /// calling EnableAccelerationAsync.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        const string bucketName = "doc-example-bucket";

        await EnableAccelerationAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method sets the configuration to enable transfer acceleration
    /// for the bucket referred to in the bucketName parameter.
    /// </summary>
    /// <param name="client">An Amazon S3 client used to enable the
    /// acceleration on an Amazon S3 bucket.</param>
```

```
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
the
        /// method will be enabling acceleration.</param>
        private static async Task EnableAccelerationAsync(AmazonS3Client client,
string bucketName)
        {
            try
            {
                var putRequest = new PutBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                    AccelerateConfiguration = new AccelerateConfiguration
                    {
                        Status = BucketAccelerateStatus.Enabled,
                    },
                };
                await client.PutBucketAccelerateConfigurationAsync(putRequest);

                var getRequest = new GetBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                };
                var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

                Console.WriteLine($"Acceleration state = '{response.Status}' ");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error occurred. Message:'{ex.Message}' when
setting transfer acceleration");
            }
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「PutBucketAccelerateConfiguration」を参照してください。

CLI

AWS CLI

バケットの加速設定を指定するには

次の put-bucket-accelerate-configuration の例では、指定したバケットの加速設定を有効にします。

```
aws s3api put-bucket-accelerate-configuration \
    --bucket my-bucket \
    --accelerate-configuration Status=Enabled
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「PutBucketAccelerateConfiguration」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットの転送加速を有効にします。

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 's3testbucket' -
AccelerateConfiguration_Status $statusVal
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「PutBucketAccelerateConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketAcl** を使用する

以下のコード例は、PutBucketAcl の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- アクセスコントロールリスト (ACL) を管理する

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        /// <summary>
        /// Creates an Amazon S3 bucket with an ACL to control access to the
        /// bucket and the objects stored in it.
        /// </summary>
        /// <param name="client">The initialized client object used to create
        /// an Amazon S3 bucket, with an ACL applied to the bucket.
        /// </param>
        /// <param name="region">The AWS Region where the bucket will be
created.</param>
        /// <param name="newBucketName">The name of the bucket to create.</param>
        /// <returns>A boolean value indicating success or failure.</returns>
        public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
client, S3Region region, string newBucketName)
        {
            try
            {
                // Create a new Amazon S3 bucket with Canned ACL.
                var putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = region,
                    CannedACL = S3CannedACL.LogDeliveryWrite,
                };

                PutBucketResponse putBucketResponse = await
client.PutBucketAsync(putBucketRequest);

                return putBucketResponse.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
```

```
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Amazon S3 error: {ex.Message}");
            }

            return false;
        }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「PutBucketAcl」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
 &ownerID,
                              const Aws::String &granteePermission,
                              const Aws::String &granteeType, const Aws::String
 &granteeID,
                              const Aws::String &granteeEmailAddress,
                              const Aws::String &granteeURI, const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
```

```
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
            s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
                  << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
```

```
/*!
 \param access: Human readable string.
 \return Permission: A Permission enum.
*/

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
*/

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「PutBucketAcl」を参照してください。

CLI

AWS CLI

この例は、`full control` 権限を 2 人の AWS ユーザー (user1@example.com と user2@example.com) に付与し、`read` 権限をすべてのユーザーに付与します。

```
aws s3api put-bucket-acl --bucket MyBucket --grant-full-control
 emailaddress=user1@example.com,emailaddress=user2@example.com --grant-read
 uri=http://acs.amazonaws.com/groups/global/AllUsers
```

カスタム ACL の詳細については、http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html を参照してください (`put-bucket-acl` などの s3api ACL コマンドは、同じ略記法を使用します)。

- API の詳細については、AWS CLI コマンドリファレンスの「PutBucketAcl」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <id>\s

                Where:
                  bucketName - The Amazon S3 bucket to grant permissions on.\s
                  id - The ID of the owner of this bucket (you can get this value
 from the AWS Management Console).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                    .grantee(builder -> builder.id(id)
```

```
                        .type(Type.CANONICAL_USER))
                .permission(Permission.FULL_CONTROL)
                .build();

            List<Grant> grantList2 = new ArrayList<>();
            grantList2.add(ownerGrant);

            AccessControlPolicy acl = AccessControlPolicy.builder()
                    .owner(builder -> builder.id(id))
                    .grants(grantList2)
                    .build();

            PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                    .bucket(bucketName)
                    .accessControlPolicy(acl)
                    .build();

            s3.putBucketAcl(putAclReq);

        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「PutBucketAcl」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケット ACL をプットします。

```
import { PutBucketAclCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Most Amazon S3 use cases don't require the use of access control lists (ACLs).
// We recommend that you disable ACLs, except in unusual circumstances where
// you need to control access for each object individually.
// Consider a policy instead. For more information see https://
docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-policies.html.
export const main = async () => {
  // Grant a user READ access to a bucket.
  const command = new PutBucketAclCommand({
    Bucket: "test-bucket",
    AccessControlPolicy: {
      Grants: [
        {
          Grantee: {
            // The canonical ID of the user. This ID is an obfuscated form of
 your AWS account number.
            // It's unique to Amazon S3 and can't be found elsewhere.
            // For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/finding-canonical-user-id.html.
            ID: "canonical-id-1",
            Type: "CanonicalUser",
          },
          // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
          // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_Grant.html#AmazonS3-Type-Grant-Permission
          Permission: "FULL_CONTROL",
        },
      ],
      Owner: {
        ID: "canonical-id-2",
      },
    },
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
```

```
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「PutBucketAcl」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }
```

```
    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「PutBucketAcl」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
```

```
        self.bucket = bucket
        self.name = bucket.name


    def grant_log_delivery_access(self):
        """
        Grant the AWS Log Delivery group write access to the bucket so that
        Amazon S3 can deliver access logs to the bucket. This is the only
recommended
        use of an S3 bucket ACL.
        """
        try:
            acl = self.bucket.Acl()
            # Putting an ACL overwrites the existing ACL. If you want to preserve
            # existing grants, append new grants to the list of existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {
                        "Type": "Group",
                        "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
                    },
                    "Permission": "WRITE",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted log delivery access to bucket '%s'",
self.bucket.name)
        except ClientError:
            logger.exception("Couldn't add ACL to bucket '%s'.",
self.bucket.name)
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「PutBucketAcl」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketCors** を使用する

以下のコード例は、PutBucketCors の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        /// <summary>
        /// Add CORS configuration to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to apply the CORS configuration to an Amazon S3 bucket.</param>
        /// <param name="configuration">The CORS configuration to apply.</param>
        private static async Task PutCORSConfigurationAsync(AmazonS3Client
client, CORSConfiguration configuration)
        {
            PutCORSConfigurationRequest request = new
PutCORSConfigurationRequest()
            {
                BucketName = BucketName,
                Configuration = configuration,
            };

            _ = await client.PutCORSConfigurationAsync(request);
        }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「PutBucketCors」を参照してください。

CLI

AWS CLI

次の例は、www.example.com からの PUT、POST、および DELETE の各リクエストを有効化し、任意のドメインからの GET リクエストを有効化します。

```
aws s3api put-bucket-cors --bucket MyBucket --cors-configuration file://cors.json

cors.json:
{
  "CORSRules": [
    {
      "AllowedOrigins": ["http://www.example.com"],
      "AllowedHeaders": ["*"],
      "AllowedMethods": ["PUT", "POST", "DELETE"],
      "MaxAgeSeconds": 3000,
      "ExposeHeaders": ["x-amz-server-side-encryption"]
    },
    {
      "AllowedOrigins": ["*"],
      "AllowedHeaders": ["Authorization"],
      "AllowedMethods": ["GET"],
      "MaxAgeSeconds": 3000
    }
  ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「PutBucketCors」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <accountId>\s

                Where:
                    bucketName - The Amazon S3 bucket to upload an object into.
                    accountId - The id of the account that owns the Amazon S3
 bucket.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
```

```
                .region(region)
                .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    public static void deleteBucketCorsInformation(S3Client s3, String
bucketName, String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                    .bucket(bucketName)
                    .expectedBucketOwner(accountId)
                    .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest =
GetBucketCorsRequest.builder()
                    .bucket(bucketName)
                    .expectedBucketOwner(accountId)
                    .build();

            GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.corsRules();
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setCorsInformation(S3Client s3, String bucketName, String
 accountId) {
        List<String> allowMethods = new ArrayList<>();
        allowMethods.add("PUT");
        allowMethods.add("POST");
        allowMethods.add("DELETE");

        List<String> allowOrigins = new ArrayList<>();
        allowOrigins.add("http://example.com");
        try {
            // Define CORS rules.
            CORSRule corsRule = CORSRule.builder()
                    .allowedMethods(allowMethods)
                    .allowedOrigins(allowOrigins)
                    .build();

            List<CORSRule> corsRules = new ArrayList<>();
            corsRules.add(corsRule);
            CORSConfiguration configuration = CORSConfiguration.builder()
                    .corsRules(corsRules)
                    .build();

            PutBucketCorsRequest putBucketCorsRequest =
 PutBucketCorsRequest.builder()
                    .bucket(bucketName)
                    .corsConfiguration(configuration)
                    .expectedBucketOwner(accountId)
                    .build();

            s3.putBucketCors(putBucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「PutBucketCors」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CORS ルールを追加します。

```
import { PutBucketCorsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// By default, Amazon S3 doesn't allow cross-origin requests. Use this command
// to explicitly allow cross-origin requests.
export const main = async () => {
  const command = new PutBucketCorsCommand({
    Bucket: "test-bucket",
    CORSConfiguration: {
      CORSRules: [
        {
          // Allow all headers to be sent to this bucket.
          AllowedHeaders: ["*"],
          // Allow only GET and PUT methods to be sent to this bucket.
          AllowedMethods: ["GET", "PUT"],
          // Allow only requests from the specified origin.
          AllowedOrigins: ["https://www.example.com"],
          // Allow the entity tag (ETag) header to be returned in the response.
 The ETag header
          // The entity tag represents a specific version of the object. The ETag
 reflects
          // changes only to the contents of an object, not its metadata.
          ExposeHeaders: ["ETag"],
```

```
          // How long the requesting browser should cache the preflight response.
 After
          // this time, the preflight request will have to be made again.
          MaxAgeSeconds: 3600,
        },
      ],
    },
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「PutBucketCors」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
```

```
        self.bucket = bucket
        self.name = bucket.name


    def put_cors(self, cors_rules):
        """
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that
are
        allowed from other domains.

        :param cors_rules: The CORS rules to apply.
        """
        try:
            self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
            logger.info(
                "Put CORS rules %s for bucket '%s'.", cors_rules,
self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't put CORS rules for bucket %s.",
self.bucket.name)
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「PutBucketCors」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
```

```
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「PutBucketCors」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketEncryption** を使用する

以下のコード例は、PutBucketEncryption の使用方法を示しています。

CLI

AWS CLI

バケットのサーバー側の暗号化を設定するには

次の put-bucket-encryption の例では、指定したバケットのデフォルトとして AES256 暗号化を設定します。

```
aws s3api put-bucket-encryption \
    --bucket my-bucket \
    --server-side-encryption-configuration '{"Rules":
 [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「PutBucketEncryption」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定したバケットで Amazon S3 マネージドキー (SSE-S3) を使用したデフォルトの AES256 サーバー側暗号化を有効にします。

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 's3testbucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「PutBucketEncryption」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketLifecycleConfiguration** を使用する

以下のコード例は、PutBucketLifecycleConfiguration の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 不完全なマルチパートアップロードを特定する
- バージョン管理されたオブジェクトを操作する

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        /// <summary>
        /// Adds lifecycle configuration information to the S3 bucket named in
        /// the bucketName parameter.
        /// </summary>
        /// <param name="client">The S3 client used to call the
        /// PutLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">A string representing the S3 bucket to
        /// which configuration information will be added.</param>
        /// <param name="configuration">A LifecycleConfiguration object that
        /// will be applied to the S3 bucket.</param>
        public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
string bucketName, LifecycleConfiguration configuration)
        {
```

```
            var request = new PutLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
                Configuration = configuration,
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「PutBucketLifecycleConfiguration」を参照してください。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットにライフサイクル設定を適用します。

```
aws s3api put-bucket-lifecycle-configuration --bucket my-bucket --lifecycle-
configuration  file://lifecycle.json
```

lifecycle.json ファイルは、現在のフォルダ内の JSON ドキュメントで、次の 2 つのルールを指定します。

```
{
    "Rules": [
        {
            "ID": "Move rotated logs to Glacier",
            "Prefix": "rotated/",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Date": "2015-11-10T00:00:00.000Z",
                    "StorageClass": "GLACIER"
                }
            ]
        },
        {
            "Status": "Enabled",
            "Prefix": "",
            "NoncurrentVersionTransitions": [
```

```
                {
                    "NoncurrentDays": 2,
                    "StorageClass": "GLACIER"
                }
            ],
            "ID": "Move old versions to Glacier"
        }
    ]
}
```

1 つ目のルールは、指定した日付に、プレフィックス rotated の付いたファイルを Glacier に移動します。2 つ目のルールは、最新でない古いオブジェクトバージョンを Glacier に移動します。詳細については、「AWS CLI ユーザーガイド」の「Specifying Parameter Values」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「PutBucketLifecycleConfiguration」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
```

```
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class LifecycleConfiguration {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                  <bucketName> <accountId>\s

                                Where:
                                  bucketName - The Amazon Simple Storage Service
 (Amazon S3) bucket to upload an object into.
                                  accountId - The id of the account that owns the
 Amazon S3 bucket.
                                """;

                if (args.length != 2) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String bucketName = args[0];
                String accountId = args[1];
                Region region = Region.US_EAST_1;
                S3Client s3 = S3Client.builder()
                                .region(region)
                                .build();

                setLifecycleConfig(s3, bucketName, accountId);
                getLifecycleConfig(s3, bucketName, accountId);
```

```
                deleteLifecycleConfig(s3, bucketName, accountId);
                System.out.println("You have successfully created, updated, and
deleted a Lifecycle configuration");
                s3.close();
        }

        public static void setLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
                try {
                        // Create a rule to archive objects with the
"glacierobjects/" prefix to Amazon
                        // S3 Glacier.
                        LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                                        .prefix("glacierobjects/")
                                        .build();

                        Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                        .days(0)
                                        .build();

                        LifecycleRule rule1 = LifecycleRule.builder()
                                        .id("Archive immediately rule")
                                        .filter(ruleFilter)
                                        .transitions(transition)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Create a second rule.
                        Transition transition2 = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                        .days(0)
                                        .build();

                        List<Transition> transitionList = new ArrayList<>();
                        transitionList.add(transition2);

                        LifecycleRuleFilter ruleFilter2 =
LifecycleRuleFilter.builder()
                                        .prefix("glacierobjects/")
                                        .build();
```

```
                        LifecycleRule rule2 = LifecycleRule.builder()
                                        .id("Archive and then delete rule")
                                        .filter(ruleFilter2)
                                        .transitions(transitionList)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Add the LifecycleRule objects to an ArrayList.
                        ArrayList<LifecycleRule> ruleList = new ArrayList<>();
                        ruleList.add(rule1);
                        ruleList.add(rule2);

                        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                                        .rules(ruleList)
                                        .build();

                        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                                        .builder()
                                        .bucket(bucketName)

.lifecycleConfiguration(lifecycleConfiguration)
                                        .expectedBucketOwner(accountId)
                                        .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }

        // Retrieve the configuration and add a new rule.
        public static void getLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
                try {
                        GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
                                        .builder()
                                        .bucket(bucketName)
```

```
                                        .expectedBucketOwner(accountId)
                                        .build();

                        GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
                        List<LifecycleRule> newList = new ArrayList<>();
                        List<LifecycleRule> rules = response.rules();
                        for (LifecycleRule rule : rules) {
                                newList.add(rule);
                        }

                        // Add a new rule with both a prefix predicate and a tag
predicate.
                        LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                                        .prefix("YearlyDocuments/")
                                        .build();

                        Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                        .days(3650)
                                        .build();

                        LifecycleRule rule1 = LifecycleRule.builder()
                                        .id("NewRule")
                                        .filter(ruleFilter)
                                        .transitions(transition)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Add the new rule to the list.
                        newList.add(rule1);
                        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                                        .rules(newList)
                                        .build();

                        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                                        .builder()
                                        .bucket(bucketName)
```

```
 .lifecycleConfiguration(lifecycleConfiguration)
                                          .expectedBucketOwner(accountId)
                                          .build();


 s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }

        // Delete the configuration from the Amazon S3 bucket.
        public static void deleteLifecycleConfig(S3Client s3, String bucketName,
 String accountId) {
                try {
                        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest
 = DeleteBucketLifecycleRequest
                                          .builder()
                                          .bucket(bucketName)
                                          .expectedBucketOwner(accountId)
                                          .build();

                        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「PutBucketLifecycleConfiguration」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_lifecycle_configuration(self, lifecycle_rules):
        """
        Apply a lifecycle configuration to the bucket. The lifecycle
 configuration can
        be used to archive or delete the objects in the bucket according to
 specified
        parameters, such as a number of days.

        :param lifecycle_rules: The lifecycle rules to apply.
        """
        try:
            self.bucket.LifecycleConfiguration().put(
                LifecycleConfiguration={"Rules": lifecycle_rules}
            )
            logger.info(
                "Put lifecycle rules %s for bucket '%s'.",
                lifecycle_rules,
                self.bucket.name,
            )
```

```
        except ClientError:
            logger.exception(
                "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「PutBucketLifecycleConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketLogging** を使用する

以下のコード例は、PutBucketLogging の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
```

```
    /// logs.
    /// </summary>
    public class ServerAccessLogging
    {
        private static IConfiguration _configuration = null!;

        public static async Task Main()
        {
            LoadConfig();

            string bucketName = _configuration["BucketName"];
            string logBucketName = _configuration["LogBucketName"];
            string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
            string accountId = _configuration["AccountId"];

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Update bucket policy for target bucket to allow delivery of
logs to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }
```

```
        /// <summary>
        /// This method grants appropriate permissions for logging to the
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" +
logPrefix + @"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3:::" + sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
```

```
            Console.WriteLine(newPolicy);

            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be
 stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
 used
        /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
 you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
 logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
```

```
            {
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load
 local settings.
                .Build();
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「PutBucketLogging」を参照してください。

CLI

AWS CLI

例 1: バケットポリシーのログ記録を設定するには

次の put-bucket-logging の例では、MyBucket のログ記録ポリシーを設定します。まず、バケットポリシーを使用して、ログ記録サービスプリンシパルに許可を付与します。

```
aws s3api put-bucket-policy \
    --bucket MyBucket \
    --policy file://policy.json
```

policy.json の内容:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {"Service": "logging.s3.amazonaws.com"},
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::MyBucket/Logs/*",
            "Condition": {
                "ArnLike": {"aws:SourceARN": "arn:aws:s3:::SOURCE-BUCKET-NAME"},
                "StringEquals": {"aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"}
            }
        }
    ]
}
```

ログ記録ポリシーを適用するには、put-bucket-logging を使用します。

```
aws s3api put-bucket-logging \
    --bucket MyBucket \
    --bucket-logging-status file://logging.json
```

logging.json の内容:

```
{
     "LoggingEnabled": {
         "TargetBucket": "MyBucket",
         "TargetPrefix": "Logs/"
     }
 }
```

put-bucket-policy コマンドは、ログ記録サービスプリンシパルに s3:PutObject アクセス許可を付与するために必要です。

詳細については、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 サーバーアクセスのログ記録」を参照してください。

例 2: 1 人のユーザーのみにアクセスをログ記録するためのバケットポリシーを設定するには

次の put-bucket-logging の例では、MyBucket のログ記録ポリシーを設定します。AWS ユーザー bob@example.com はログファイルを完全に制御でき、他のユーザーはアクセスできません。まず、put-bucket-acl で S3 アクセス許可を付与します。

```
aws s3api put-bucket-acl \
    --bucket MyBucket \
    --grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery \
    --grant-read-acp URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

次に、put-bucket-logging を使用してログ記録ポリシーを適用します。

```
aws s3api put-bucket-logging \
    --bucket MyBucket \
    --bucket-logging-status file://logging.json
```

logging.json の内容:

```
{
    "LoggingEnabled": {
        "TargetBucket": "MyBucket",
        "TargetPrefix": "MyBucketLogs/",
        "TargetGrants": [
            {
                "Grantee": {
                    "Type": "AmazonCustomerByEmail",
                    "EmailAddress": "bob@example.com"
                },
                "Permission": "FULL_CONTROL"
            }
        ]
    }
}
```

put-bucket-acl コマンドは、Amazon S3 のログ配信システムに必要なアクセス許可 (write および read-acp アクセス許可) を付与するために必要です。

詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「Amazon S3 サーバーアクセスログ」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「PutBucketLogging」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketNotification** を使用する

以下のコード例は、PutBucketNotification の使用方法を示しています。

CLI

AWS CLI

my-bucket という名前のバケットに通知設定を適用します。

```
aws s3api put-bucket-notification --bucket my-bucket --notification-configuration
 file://notification.json
```

notification.json ファイルは、現在のフォルダにある JSON ファイルで、モニタリングする SNS トピックとイベントタイプを指定します。

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

SNS トピックには、Amazon S3 に公開を許可する IAM ポリシーがアタッチされている必要があります。

```
{
 "Version": "2008-10-17",
 "Id": "example-ID",
 "Statement": [
  {
   "Sid": "example-statement-ID",
   "Effect": "Allow",
   "Principal": {
     "Service": "s3.amazonaws.com"
   },
   "Action": [
```

```
    "SNS:Publish"
   ],
   "Resource": "arn:aws:sns:us-west-2:123456789012:my-bucket",
   "Condition": {
      "ArnLike": {
      "aws:SourceArn": "arn:aws:s3:*:*:my-bucket"
    }
   }
  }
 ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「PutBucketNotification」を参照してください。

PowerShell

Tools for PowerShell

例 1: この例では、S3 イベント ObjectRemovedDelete の SNS トピック設定を設定して、指定した S3 バケットの通知を有効にします。

```
$topic =  [Amazon.S3.Model.TopicConfiguration] @{
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName kt-tools -TopicConfiguration $topic
```

例 2: この例では、指定したバケットの ObjectCreatedAll の通知を有効にして、Lambda 関数に送信します。

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
  Id = "ObjectCreated-Lambda"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".pem"}
```

```
      )
    }
  }
}

Write-S3BucketNotification -BucketName ssm-editor -LambdaFunctionConfiguration
 $lambdaConfig
```

例 3: この例では、異なるキーサフィックスに基づいて 2 つの異なる Lambda 設定を作成し、両方を 1 つのコマンドで設定します。

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
  Id = "ObjectCreated-dada-ps1"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".ps1"}
      )
    }
  }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}
```

```
Write-S3BucketNotification -BucketName ssm-editor -LambdaFunctionConfiguration
 $firstLambdaConfig,$secondlambdaConfig
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンスの「PutBucketNotification」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketNotificationConfiguration** を使用する

以下のコード例は、PutBucketNotificationConfiguration の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to enable notifications for an Amazon Simple
/// Storage Service (Amazon S3) bucket.
/// </summary>
public class EnableNotifications
{
    public static async Task Main()
    {
        const string bucketName = "doc-example-bucket1";
```

```
            const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
            const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";

            IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
            await EnableNotificationAsync(client, bucketName, snsTopic,
 sqsQueue);
        }

        /// <summary>
        /// This method makes the call to the PutBucketNotificationAsync method.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to call
        /// the PutBucketNotificationAsync method.</param>
        /// <param name="bucketName">The name of the bucket for which
        /// notifications will be turned on.</param>
        /// <param name="snsTopic">The ARN for the Amazon Simple Notification
        /// Service (Amazon SNS) topic associated with the S3 bucket.</param>
        /// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
        /// (Amazon SQS) queue to which notifications will be pushed.</param>
        public static async Task EnableNotificationAsync(
            IAmazonS3 client,
            string bucketName,
            string snsTopic,
            string sqsQueue)
        {
            try
            {
                // The bucket for which we are setting up notifications.
                var request = new PutBucketNotificationRequest()
                {
                    BucketName = bucketName,
                };

                // Defines the topic to use when sending a notification.
                var topicConfig = new TopicConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic,
                };
                request.TopicConfigurations = new List<TopicConfiguration>
                {
                    topicConfig,
```

```
                };
                request.QueueConfigurations = new List<QueueConfiguration>
                {
                    new QueueConfiguration()
                    {
                        Events = new List<EventType>
{ EventType.ObjectCreatedPut },
                        Queue = sqsQueue,
                    },
                };

                // Now apply the notification settings to the bucket.
                PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「PutBucketNotificationConfiguration」を参照してください。

CLI

AWS CLI

バケットへの指定された通知を有効にするには

次の put-bucket-notification-configuration の例では、my-bucket という名前のバケットに通知設定を適用します。notification.json ファイルは、現在のフォルダにある JSON ファイルで、モニタリングする SNS トピックとイベントタイプを指定します。

```
aws s3api put-bucket-notification-configuration \
    --bucket my-bucket \
    --notification-configuration file://notification.json
```

notification.json の内容:

```
{
    "TopicConfigurations": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-
topic",
            "Events": [
                "s3:ObjectCreated:*"
            ]
        }
    ]
}
```

SNS トピックには、Amazon S3 に公開を許可する IAM ポリシーがアタッチされている必要があります。

```
{
    "Version": "2008-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "arn:aws:sns:us-west-2:123456789012::s3-notification-
topic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:my-bucket"
                }
            }
        }
    ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「PutBucketNotificationConfiguration」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Event;
import software.amazon.awssdk.services.s3.model.NotificationConfiguration;
import
 software.amazon.awssdk.services.s3.model.PutBucketNotificationConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.TopicConfiguration;
import java.util.ArrayList;
import java.util.List;

public class SetBucketEventBridgeNotification {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The Amazon S3 bucket.\s
                    topicArn - The Simple Notification Service topic ARN.\s
                    id - An id value used for the topic configuration. This value
 is displayed in the AWS Management Console.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String topicArn = args[1];
```

```
        String id = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        setBucketNotification(s3Client, bucketName, topicArn, id);
        s3Client.close();
    }

    public static void setBucketNotification(S3Client s3Client, String
bucketName, String topicArn, String id) {
        try {
            List<Event> events = new ArrayList<>();
            events.add(Event.S3_OBJECT_CREATED_PUT);

            TopicConfiguration config = TopicConfiguration.builder()
                    .topicArn(topicArn)
                    .events(events)
                    .id(id)
                    .build();

            List<TopicConfiguration> topics = new ArrayList<>();
            topics.add(config);

            NotificationConfiguration configuration =
NotificationConfiguration.builder()
                    .topicConfigurations(topics)
                    .build();

            PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
                    .builder()
                    .bucket(bucketName)
                    .notificationConfiguration(configuration)
                    .skipDestinationValidation(true)
                    .build();

            // Set the bucket notification configuration.
            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

        } catch (S3Exception e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「PutBucketNotificationConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketPolicy** を使用する

以下のコード例は、PutBucketPolicy の使用方法を示しています。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                 const Aws::String &policyBody,
                                 const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
            Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);
```

```
    Aws::S3::Model::PutBucketPolicyOutcome outcome =
            s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}


//! Build a policy JSON string.
/*!
  \param userArn: Aws user Amazon Resource Name (ARN).
      For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
  \param bucketName: Name of a bucket.
  \return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                            const Aws::String &bucketName) {
    return
            "{\n"
            "   \"Version\":\"2012-10-17\",\n"
            "   \"Statement\":[\n"
            "       {\n"
            "           \"Sid\": \"1\",\n"
            "           \"Effect\": \"Allow\",\n"
            "           \"Principal\": {\n"
            "               \"AWS\": \""
            + userArn +
            "\"\n""          },\n"
            "           \"Action\": [ \"s3:getObject\" ],\n"
            "           \"Resource\": [ \"arn:aws:s3:::"
            + bucketName +
            "/*\" ]\n"
            "       }\n"
```

```
            "   ]\n"
            "}";
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「PutBucketPolicy」を参照してください。

CLI

AWS CLI

この例では、MySecretFolder 内のものを除き、すべてのユーザーが MyBucket 内のすべてのオブジェクトを取得できるようにします。また、AWS アカウント 1234-5678-9012 のルートユーザーに、put および delete 権限を付与します。

```
aws s3api put-bucket-policy --bucket MyBucket --policy file://policy.json

policy.json:
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": "*",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::MyBucket/*"
      },
      {
         "Effect": "Deny",
         "Principal": "*",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::MyBucket/MySecretFolder/*"
      },
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:root"
         },
         "Action": [
            "s3:DeleteObject",
            "s3:PutObject"
         ],
```

```
            "Resource": "arn:aws:s3:::MyBucket/*"
        }
    ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「PutBucketPolicy」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                    <bucketName> <polFile>

                Where:
                    bucketName - The Amazon S3 bucket to set the policy on.
                    polFile - A JSON file containing the policy (see the Amazon
S3 Readme for an example).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }

    public static void setPolicy(S3Client s3, String bucketName, String
policyText) {
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();

            s3.putBucketPolicy(policyReq);
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }

    // Loads a JSON-formatted policy from a file
    public static String getBucketPolicyFromFile(String policyFile) {

        StringBuilder fileText = new StringBuilder();
        try {
            List<String> lines = Files.readAllLines(Paths.get(policyFile),
 StandardCharsets.UTF_8);
            for (String line : lines) {
                fileText.append(line);
            }

        } catch (IOException e) {
            System.out.format("Problem reading file: \"%s\"", policyFile);
            System.out.println(e.getMessage());
        }

        try {
            final JsonParser parser = new
 ObjectMapper().getFactory().createParser(fileText.toString());
            while (parser.nextToken() != null) {
            }

        } catch (IOException jpe) {
            jpe.printStackTrace();
        }
        return fileText.toString();
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「PutBucketPolicy」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

ポリシーを追加します。

```
import { PutBucketPolicyCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new PutBucketPolicyCommand({
    Policy: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Sid: "AllowGetObject",
          // Allow this particular user to call GetObject on any object in this
 bucket.
          Effect: "Allow",
          Principal: {
            AWS: "arn:aws:iam::ACCOUNT-ID:user/USERNAME",
          },
          Action: "s3:GetObject",
          Resource: "arn:aws:s3:::BUCKET-NAME/*",
        },
      ],
    }),
    // Apply the preceding policy to this bucket.
    Bucket: "BUCKET-NAME",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
```

```
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「PutBucketPolicy」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_policy(self, policy):
        """
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
```

```
            logger.info("Put policy %s for bucket '%s'.", policy,
 self.bucket.name)
        except ClientError:
            logger.exception("Couldn't apply policy to bucket '%s'.",
 self.bucket.name)
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「PutBucketPolicy」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def set_policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
```

```
  end

end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「PutBucketPolicy」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketReplication** を使用する

以下のコード例は、PutBucketReplication の使用方法を示しています。

CLI

AWS CLI

S3 バケットのレプリケーションを設定するには

次の put-bucket-replication の例では、指定した S3 バケットにレプリケーション設定を適用します。

```
aws s3api put-bucket-replication \
    --bucket AWSDOC-EXAMPLE-BUCKET1 \
    --replication-configuration file://replication.json
```

replication.json の内容:

```
{
    "Role": "arn:aws:iam::123456789012:role/s3-replication-role",
    "Rules": [
        {
            "Status": "Enabled",
            "Priority": 1,
            "DeleteMarkerReplication": { "Status": "Disabled" },
            "Filter" : { "Prefix": ""},
            "Destination": {
                "Bucket": "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET2"
            }
```

```
        }
    ]
}
```

レプリケーション先のバケットではバージョニングが有効になっている必要があります。指定したロールは、レプリケーション先のバケットへの書き込みアクセス許可を持ち、Amazon S3 がそのロールを引き受けることを許可する信頼関係が必要です。

ロールのアクセス許可ポリシーの例:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetReplicationConfiguration",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET1/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ReplicateObject",
                "s3:ReplicateDelete",
                "s3:ReplicateTags"
            ],
            "Resource": "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET2/*"
        }
```

```
    ]
}
```

信頼関係ポリシーの例:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

このコマンドでは何も出力されません。

詳細については、「Amazon Simple Storage Service コンソールユーザーガイド」の「チュートリアル: レプリケーションの設定例」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「PutBucketReplication」を参照してください。

PowerShell

Tools for PowerShell

例 1: この例では、バケット「examplebucket」内のキー名プレフィックス「TaxDocs」で作成された新しいオブジェクトの「exampletargetbucket」バケットへのレプリケーションを有効にする単一のルールを使用してレプリケーション設定を指定します。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
```

```
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

例 2: この例では、バケット「examplebucket」内のキー名プレフィックス「TaxDocs」または「OtherDocs」で作成された新しいオブジェクトの「exampletargetbucket」バケットへのレプリケーションを有効にする複数のルールを使用してレプリケーション設定を指定します。キーのプレフィックスの重複は許可されません。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

例 3: この例では、指定したバケットのレプリケーション設定を更新して、キー名プレフィックス「TaxDocs」を持つオブジェクトのバケット「exampletargetbucket」へのレプリケーションを制御するルールを無効にします。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }
```

```
$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「PutBucketReplication」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketRequestPayment** を使用する

以下のコード例は、PutBucketRequestPayment の使用方法を示しています。

CLI

AWS CLI

例 1: バケットの「リクエスタ支払い」設定を有効にするには

次の put-bucket-request-payment の例では、指定したバケットの requester pays を有効にします。

```
aws s3api put-bucket-request-payment \
    --bucket my-bucket \
    --request-payment-configuration '{"Payer":"Requester"}'
```

このコマンドでは何も出力されません。

例 2: バケットの「リクエスタ支払い」設定を無効にするには

次の put-bucket-request-payment の例では、指定したバケットの requester pays を無効にします。

```
aws s3api put-bucket-request-payment \
```

```
    --bucket my-bucket \
    --request-payment-configuration '{"Payer":"BucketOwner"}'
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「PutBucketRequestPayment」を参照してください。

PowerShell

Tools for PowerShell

例 1: 「mybucket」という名前のバケットのリクエスト支払い設定を更新して、バケットからのダウンロードをリクエストしたユーザーにダウンロード料金が請求されるようにします。デフォルトでは、バケット所有者がダウンロード料金を支払います。リクエストの支払いをデフォルトに戻すには、RequestPaymentConfiguration_Payer パラメーターに「BucketOwner」を使用します。

```
Write-S3BucketRequestPayment -BucketName mybucket -
RequestPaymentConfiguration_Payer Requester
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「PutBucketRequestPayment」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketTagging** を使用する

以下のコード例は、PutBucketTagging の使用方法を示しています。

CLI

AWS CLI

次のコマンドは、タグ付けの設定を my-bucket という名前のバケットに適用します。

```
aws s3api put-bucket-tagging --bucket my-bucket --tagging file://tagging.json
```

この tagging.json ファイルは、タグを指定する現在のフォルダ内の JSON ドキュメントです。

```
{
   "TagSet": [
     {
       "Key": "organization",
       "Value": "marketing"
     }
   ]
}
```

または、コマンドラインから直接タグ設定を my-bucket に適用します。

```
aws s3api put-bucket-tagging --bucket my-bucket --tagging
 'TagSet=[{Key=organization,Value=marketing}]'
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「PutBucketTagging」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、**cloudtrail-test-2018** という名前のバケットに 2 つのタグを適用します。1 つは Stage のキーと値が Test のタグで、もう 1 つはキーが Environment で、値は Alpha のタグです。タグがバケットに追加されたことを確認するには、**Get-S3BucketTagging -BucketName bucket_name** を実行します。結果では、最初のコマンドでバケットに適用したタグが表示されるはずです。**Write-S3BucketTagging** は、バケットに設定されている既存のタグセット全体を上書きすることに注意が必要です。個別のタグを追加または削除するには、リソースグループとタグ付けの API コマンドレット、**Add-RGTResourceTag**、**Remove-RGTResourceTag** を実行します。または、AWS マネジメントコンソールのタグエディタを使用して S3 バケットのタグを管理します。

```
Write-S3BucketTagging -BucketName cloudtrail-test-2018 -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

例 2: このコマンドは、バケット **cloudtrail-test-2018** を **Write-S3BucketTagging** コマンドレットにパイプします。これにより、Stage:Production と Department:Finance とい

うタグがバケットに適用されます。**Write-S3BucketTagging** は、バケットに設定されている既存のタグセット全体を上書きすることに注意が必要です。

```
Get-S3Bucket -BucketName cloudtrail-test-2018 | Write-S3BucketTagging
 -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
 Value="Finance" } )
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「PutBucketTagging」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketVersioning** を使用する

以下のコード例は、PutBucketVersioning の使用方法を示しています。

CLI

AWS CLI

次のコマンドは、my-bucket という名前のバケットのバージョニングを有効にします。

```
aws s3api put-bucket-versioning --bucket my-bucket --versioning-configuration
 Status=Enabled
```

次のコマンドはバージョニングを有効にします。これは、MFA コードを使用します。

```
aws s3api put-bucket-versioning --bucket my-bucket --versioning-configuration
 Status=Enabled --mfa "SERIAL 123456"
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「PutBucketVersioning」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケットのバージョニングを有効にします。

```
Write-S3BucketVersioning -BucketName 's3testbucket' -VersioningConfig_Status
 Enabled
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「PutBucketVersioning」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketWebsite** を使用する

以下のコード例は、PutBucketWebsite の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
                // Put the website configuration.
                PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument,
                    },
                };
                PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「PutBucketWebsite」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::String &indexPage, const Aws::String
 &errorPage,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
            client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
```

```
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「PutBucketWebsite」を参照してください。

CLI

AWS CLI

my-bucket という名前のバケットに静的なウェブサイト設定を適用します。

```
aws s3api put-bucket-website --bucket my-bucket --website-configuration file://
website.json
```

website.json ファイルは現在のフォルダ内の JSON ドキュメントで、ウェブサイトのインデックスページとエラーページを指定します。

```
{
    "IndexDocument": {
        "Suffix": "index.html"
    },
    "ErrorDocument": {
        "Key": "error.html"
    }
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「PutBucketWebsite」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucketName> [indexdoc]\s

                Where:
                   bucketName   - The Amazon S3 bucket to set the website
 configuration on.\s
                   indexdoc - The index document, ex. 'index.html'
                              If not specified, 'index.html' will be set.
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }

    public static void setWebsiteConfig(S3Client s3, String bucketName, String
 indexDoc) {
        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()

 .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                    .build();

            PutBucketWebsiteRequest pubWebsiteReq =
 PutBucketWebsiteRequest.builder()
                    .bucket(bucketName)
                    .websiteConfiguration(websiteConfig)
                    .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「PutBucketWebsite」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

ウェブサイト設定を設定します。

```
import { PutBucketWebsiteCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Set up a bucket as a static website.
// The bucket needs to be publicly accessible.
export const main = async () => {
  const command = new PutBucketWebsiteCommand({
    Bucket: "test-bucket",
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
      },
      IndexDocument: {
        // A suffix that is appended to a request that is for a directory.
        Suffix: "index.html",
      },
    },
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「PutBucketWebsite」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定したバケットのウェブサイトのホスティングを有効にして、インデックスドキュメントを「index.html」、エラードキュメントを「error.html」と指定します。

```
Write-S3BucketWebsite -BucketName 's3testbucket' -
WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「PutBucketWebsite」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
 configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end
```

```
  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the
 website.
  # @param error_document [String] The name of the error document to show for 4XX
 errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
 false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「PutBucketWebsite」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutObject** を使用する

以下のコード例は、PutObject の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- バケットとオブジェクトの使用を開始する
- アップロードとダウンロードを追跡する

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The Amazon S3 bucket to which the object
/// will be uploaded.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object
/// on the local computer to upload.</param>
/// <returns>A boolean value indicating the success or failure of the
/// upload procedure.</returns>
public static async Task<bool> UploadFileAsync(
    IAmazonS3 client,
    string bucketName,
```

```
            string objectName,
            string filePath)
        {
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await client.PutObjectAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully uploaded {objectName} to
{bucketName}.");
                return true;
            }
            else
            {
                Console.WriteLine($"Could not upload {objectName} to
{bucketName}.");
                return false;
            }
        }
```

サーバー側の暗号化を使用してオブジェクトをアップロードします。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket with server-side encryption enabled.
    /// </summary>
    public class ServerSideEncryption
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
```

```
            string keyName = "samplefile.txt";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            await WritingAnObjectAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// Upload a sample object include a setting for encryption.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// to upload a file and apply server-side encryption.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// encrypted object will reside.</param>
        /// <param name="keyName">The name for the object that you want to
        /// create in the supplied bucket.</param>
        public static async Task WritingAnObjectAsync(IAmazonS3 client, string
bucketName, string keyName)
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod =
ServerSideEncryptionMethod.AES256,
                };

                var putResponse = await client.PutObjectAsync(putRequest);

                // Determine the encryption state of an object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
```

```
                GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;

                Console.WriteLine($"Encryption method used: {0}",
objectEncryption.ToString());
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}' when writing an
object");
            }
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「PutObject」を参照してください。

Bash

Bash スクリプトを使用した AWS CLI

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
#############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
```

```
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「PutObject」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
 needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          fileName.c_str(),
                                          std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3Client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
```

```
        std::cout << "Added object '" << fileName << "' to bucket '"
                  << bucketName << "'.";
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「PutObject」を参照してください。

CLI

AWS CLI

次の例は、put-object コマンドを使用して、オブジェクトを Amazon S3 にアップロードします。

```
aws s3api put-object --bucket text-content --key dir-1/my_images.tar.bz2 --body
 my_images.tar.bz2
```

次の例は、動画ファイルのアップロードを示しています (動画ファイルは Windows ファイルシステム構文を使用して指定します)。

```
aws s3api put-object --bucket text-content --key dir-1/big-video-file.mp4 --body
 e:\media\videos\f-sharp-3-data-services.mp4
```

オブジェクトのアップロードの詳細については、「Amazon S3 ユーザーガイド」の「オブジェクトのアップロード」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「PutObject」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

低レベル API を使用してオブジェクトをバケットに配置します。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(bucketName string, objectKey string,
 fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(context.TODO(), &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
    fileName, bucketName, objectKey, err)
  }
 }
```

```
 return err
}
```

Transfer Manager を使用してオブジェクトをバケットにアップロードします。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
 string, contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
 bucket)
  } else {
   outKey = *output.Key
```

```
  }
 }
 return outKey, err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「PutObject」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

S3Client を使用してバケットにファイルをアップロードします。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class PutObject {
```

```
   public static void main(String[] args) {
       final String usage = """

               Usage:
                 <bucketName> <objectKey> <objectPath>\s

               Where:
                 bucketName - The Amazon S3 bucket to upload an object into.
                 objectKey - The object to upload (for example, book.pdf).
                 objectPath - The path where the file is located (for example,
C:/AWS/book2.pdf).\s
               """;

       if (args.length != 3) {
           System.out.println(usage);
           System.exit(1);
       }

       String bucketName = args[0];
       String objectKey = args[1];
       String objectPath = args[2];
       Region region = Region.US_EAST_1;
       S3Client s3 = S3Client.builder()
               .region(region)
               .build();

       putS3Object(s3, bucketName, objectKey, objectPath);
       s3.close();
   }

   // This example uses RequestBody.fromFile to avoid loading the whole file
into
   // memory.
   public static void putS3Object(S3Client s3, String bucketName, String
objectKey, String objectPath) {
       try {
           Map<String, String> metadata = new HashMap<>();
           metadata.put("x-amz-meta-myVal", "test");
           PutObjectRequest putOb = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   .metadata(metadata)
                   .build();
```

```
            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
 " + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

S3TransferManager を使用してバケットにファイルをアップロードします。完全なファイルとテストを表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String
 bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

S3Client を使用してオブジェクトをバケットにアップロードし、タグを設定します。

```
   public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
       try {
           Tag tag1 = Tag.builder()
                   .key("Tag 1")
                   .value("This is tag 1")
                   .build();

           Tag tag2 = Tag.builder()
                   .key("Tag 2")
                   .value("This is tag 2")
                   .build();

           List<Tag> tags = new ArrayList<>();
           tags.add(tag1);
           tags.add(tag2);

           Tagging allTags = Tagging.builder()
                   .tagSet(tags)
                   .build();

           PutObjectRequest putOb = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   .tagging(allTags)
                   .build();

           s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));

       } catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
       try {
           GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
                   .bucket(bucketName)
```

```
                    .key(objectKey)
                    .build();

            GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
            List<Tag> obTags = getTaggingRes.tagSet();
            for (Tag sinTag : obTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

            // Replace the object's tags with two new tags.
            Tag tag3 = Tag.builder()
                    .key("Tag 3")
                    .value("This is tag 3")
                    .build();

            Tag tag4 = Tag.builder()
                    .key("Tag 4")
                    .value("This is tag 4")
                    .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag3);
            tags.add(tag4);

            Tagging updatedTags = Tagging.builder()
                    .tagSet(tags)
                    .build();

            PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .tagging(updatedTags)
                    .build();

            s3.putObjectTagging(taggingRequest1);
            GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
            List<Tag> modTags = getTaggingRes2.tagSet();
            for (Tag sinTag : modTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
```

```
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Return a byte array.
    private static byte[] getObjectFile(String filePath) {
        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;

        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        return bytesArray;
    }
}
```

S3Client を使用してオブジェクトをバケットにアップロードし、メタデータを設定します。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = """

                Usage:
                  <bucketName> <objectKey> <objectPath>\s

                Where:
                  bucketName - The Amazon S3 bucket to upload an object into.
                  objectKey - The object to upload (for example, book.pdf).
                  objectPath - The path where the file is located (for example,
 C:/AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
 bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
```

```
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file
 into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String
 objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .metadata(metadata)
                    .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
 " + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

S3Client を使用してオブジェクトをバケットにアップロードし、オブジェクトの保持値を設定します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <key> <bucketName>\s

                Where:
                    key - The name of the object (for example, book.pdf).\s
                    bucketName - The Amazon S3 bucket name that contains the
 object (for example, bucket1).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }

    public static void setRentionPeriod(S3Client s3, String key, String bucket) {
        try {
            LocalDate localDate = LocalDate.parse("2020-07-17");
            LocalDateTime localDateTime = localDate.atStartOfDay();
```

```
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            ObjectLockRetention lockRetention = ObjectLockRetention.builder()
                    .mode("COMPLIANCE")
                    .retainUntilDate(instant)
                    .build();

            PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
                    .bucket(bucket)
                    .key(key)
                    .bypassGovernanceRetention(true)
                    .retention(lockRetention)
                    .build();

            // To set Retention on an object, the Amazon S3 bucket must support
object
            // locking, otherwise an exception is thrown.
            s3.putObjectRetention(retentionRequest);
            System.out.print("An object retention configuration was successfully
placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「PutObject」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトをアップロードします。

```
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new PutObjectCommand({
    Bucket: "test-bucket",
    Key: "hello-s3.txt",
    Body: "Hello S3!",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「PutObject」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
```

```
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「PutObject」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトをバケットにアップロードします。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
```

```
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「PutObject」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、単一のファイル「local-sample.txt」を Amazon S3 にアップロードして、バケット「test-files」に「sample.txt」というキーを持つオブジェクトを作成します。

```
Write-S3Object -BucketName test-files -Key "sample.txt" -File .\local-sample.txt
```

例 2: このコマンドは、単一のファイル「sample.txt」を Amazon S3 にアップロードして、バケット「test-files」に「sample.txt」というキーを持つオブジェクトを作成します。-Key パラメータを指定しない場合、ファイル名が S3 オブジェクトキーとして使用されます。

```
Write-S3Object -BucketName test-files -File .\sample.txt
```

例 3: このコマンドは、単一のファイル「local-sample.txt」を Amazon S3 にアップロードして、バケット「test-files」に「prefix/to/sample.txt」というキーを持つオブジェクトを作成します。

```
Write-S3Object -BucketName test-files -Key "prefix/to/sample.txt" -File .\local-
sample.txt
```

例 4: このコマンドは、サブディレクトリ「Scripts」内のすべてのファイルをバケット「test-files」にアップロードして、共通のキープレフィックス「SampleScripts」を各オブジェクトに適用します。アップロードされた各ファイルは「SampleScripts/FileName」というキーを持ちます。ただし、「filename」の部分はそれぞれ異なります。

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
```

例 5: このコマンドは、ローカルディレクトリ「Scripts」内のすべての *.ps1 ファイルをバケット「test-files」にアップロードして、共通のキープレフィックス「SampleScripts」を各オブジェクトに適用します。アップロードされた各ファイルは「SampleScripts/filename.ps1」というキーを持ちます。ただし、「filename」の部分はそれぞれ異なります。

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
 -SearchPattern *.ps1
```

例 6: このコマンドは、「sample.txt」というキーを持つ、指定されたコンテンツ文字列を含む新しい S3 オブジェクトを作成します。

```
Write-S3Object -BucketName test-files -Key "sample.txt" -Content "object
 contents"
```

例 7: このコマンドは、指定したファイル (ファイル名をキーとして使用) をアップロードして、指定したタグを新しいオブジェクトに適用します。

```
Write-S3Object -BucketName test-files -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

例 8: このコマンドは、指定したフォルダを再帰的にアップロードして、指定したタグをすべての新しいオブジェクトに適用します。

```
Write-S3Object -BucketName test-files -Folder . -KeyPrefix "TaggedFiles" -Recurse
 -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「PutObject」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string.
 When this
                     argument is a string, it is interpreted as a file name,
 which is
                     opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
            try:
                put_data = open(data, "rb")
            except IOError:
                logger.exception("Expected file name or binary data, got '%s'.",
 data)
                raise

        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'.",
```

```
                self.object.key,
                self.object.bucket_name,
            )
            raise
        finally:
            if getattr(put_data, "close", None):
                put_data.close()
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「PutObject」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

マネージドアップローダー (Object.upload_file) を使用してファイルをアップロードします。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
```

```
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Object.put を使用してファイルをアップロードします。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
```

```
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Object.put を使用してファイルをアップロードし、サーバー側の暗号化を追加します。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end
```

```
# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「PutObject」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
pub async fn upload_object(
    client: &Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<PutObjectOutput, SdkError<PutObjectError>> {
    let body = ByteStream::from_path(Path::new(file_name)).await;
    client
        .put_object()
```

```
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「PutObject」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    "Get contents of file from application server."
    DATA lv_body TYPE xstring.
    OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
    READ DATASET iv_file_name INTO lv_body.
    CLOSE DATASET iv_file_name.

    "Upload/put an object to an S3 bucket."
    TRY.
        lo_s3->putobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_file_name
            iv_body = lv_body
        ).
        MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「PutObject」を参照してください。

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

ローカルストレージからバケットにファイルをアップロードします。

```
    public func uploadFile(bucket: String, key: String, file: String) async
throws {
        let fileUrl = URL(fileURLWithPath: file)
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.from(data: fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }
```

Swift Data オブジェクトのコンテンツをバケットにアップロードします。

```
    public func createFile(bucket: String, key: String, withData data: Data)
async throws {
```

```
        let dataStream = ByteStream.from(data: data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「PutObject」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutObjectAcl** を使用する

以下のコード例は、PutObjectAcl の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- アクセスコントロールリスト (ACL) を管理する

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
 &objectKey, const Aws::String &ownerID,
```

```
                                const Aws::String &granteePermission, const
Aws::String &granteeType,
                                const Aws::String &granteeID, const Aws::String
&granteeEmailAddress,
                                const Aws::String &granteeURI, const
Aws::S3::S3ClientConfiguration &clientConfig) {
   Aws::S3::S3Client s3Client(clientConfig);

   Aws::S3::Model::Owner owner;
   owner.SetID(ownerID);

   Aws::S3::Model::Grantee grantee;
   grantee.SetType(setGranteeType(granteeType));

   if (!granteeEmailAddress.empty()) {
       grantee.SetEmailAddress(granteeEmailAddress);
   }

   if (!granteeID.empty()) {
       grantee.SetID(granteeID);
   }

   if (!granteeURI.empty()) {
       grantee.SetURI(granteeURI);
   }

   Aws::S3::Model::Grant grant;
   grant.SetGrantee(grantee);
   grant.SetPermission(setGranteePermission(granteePermission));

   Aws::Vector<Aws::S3::Model::Grant> grants;
   grants.push_back(grant);

   Aws::S3::Model::AccessControlPolicy acp;
   acp.SetOwner(owner);
   acp.SetGrants(grants);

   Aws::S3::Model::PutObjectAclRequest request;
   request.SetAccessControlPolicy(acp);
   request.SetBucket(bucketName);
   request.SetKey(objectKey);

   Aws::S3::Model::PutObjectAclOutcome outcome =
           s3Client.PutObjectAcl(request);
```

```
    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
                  << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param access: Human readable string.
 \return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration.
*/
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
```

```
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「PutObjectAcl」を参照してください。

CLI

AWS CLI

次のコマンドは、2 人の AWS ユーザー (user1@example.com および user2@example.com) に `full control` 権限を付与し、`read` 権限をすべてのユーザーに付与します。

```
aws s3api put-object-acl --bucket MyBucket --key file.txt --grant-full-control
 emailaddress=user1@example.com,emailaddress=user2@example.com --grant-read
 uri=http://acs.amazonaws.com/groups/global/AllUsers
```

カスタム ACL の詳細については、http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html を参照してください (`put-object-acl` などの s3api ACL コマンドは、同じ略記法を使用します)。

- API の詳細については、AWS CLI コマンドリファレンスの「PutObjectAcl」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class ObjectWrapper:
```

```
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put_acl(self, email):
        """
        Applies an ACL to the object that grants read access to an AWS user
identified
        by email address.

        :param email: The email address of the user to grant access.
        """
        try:
            acl = self.object.Acl()
            # Putting an ACL overwrites the existing ACL, so append new grants
            # if you want to preserve existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
email},
                    "Permission": "READ",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted read access to %s.", email)
        except ClientError:
            logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「PutObjectAcl」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutObjectLegalHold** を使用する

以下のコード例は、PutObjectLegalHold の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Amazon S3 オブジェクトをロックする

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
/// <summary>
/// Set or modify a legal hold on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="holdStatus">The On or Off status for the legal hold.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectLegalHold(string bucketName,
    string objectKey, ObjectLockLegalHoldStatus holdStatus)
{
    try
    {
        var request = new PutObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            LegalHold = new ObjectLockLegalHold()
```

```
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
 {bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「PutObjectLegalHold」を参照してください。

CLI

AWS CLI

リーガルホールドをオブジェクトに適用するには

次の put-object-legal-hold 例では、doc1.rtf という名前のバケットのオブジェクトにリーガルホールドを設定します。

```
aws s3api put-object-legal-hold \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf \
    --legal-hold Status=ON
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「PutObjectLegalHold」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
 {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }
```

```
 return err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「PutObjectLegalHold」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   // Set or modify a legal hold on an object in an S3 bucket.
   public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
       ObjectLockLegalHold legalHold ;
       if (legalHoldOn) {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.ON)
               .build();
       } else {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.OFF)
               .build();
       }

       PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
           .bucket(bucketName)
           .key(objectKey)
           .legalHold(legalHold)
           .build();

       getClient().putObjectLegalHold(legalHoldRequest) ;
```

```
        System.out.println("Modified legal hold for "+ objectKey +" in
 "+bucketName +".");
    }
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「PutObjectLegalHold」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import { PutObjectLegalHoldCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: "ON",
    },
    // Optionally, you can provide additional parameters
    // ChecksumAlgorithm: "ALGORITHM",
    // ContentMD5: "MD5_HASH",
    // ExpectedBucketOwner: "ACCOUNT_ID",
```

```
    // RequestPayer: "requester",
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const response = await client.send(command);
    console.log(
      `Object legal hold status: ${response.$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME", "OBJECT_KEY");
}
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「PutObjectLegalHold」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutObjectLockConfiguration** を使用する

以下のコード例は、PutObjectLockConfiguration の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Amazon S3 オブジェクトをロックする

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットのオブジェクトロック設定を指定します。

```
    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };
```

```
            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

バケットのデフォルトの保存期間を設定します。

```
    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
```

```
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in
days or years but not both.
                        }
                    }
                }
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「PutObjectLockConfiguration」を参照してください。

CLI

AWS CLI

バケットにオブジェクトロック設定を設定するには

次の put-object-lock-configuration 例では、指定したバケットに 50 日間のオブジェクトロックを設定します。

```
aws s3api put-object-lock-configuration \
    --bucket my-bucket-with-object-lock \
    --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule":
 { "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「PutObjectLockConfiguration」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットのオブジェクトロック設定を指定します。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
 string) error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
```

```
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

バケットのデフォルトの保存期間を設定します。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ModifyDefaultBucketRetention modifies the default retention period of an
 existing bucket.
```

```
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「PutObjectLockConfiguration」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットのオブジェクトロック設定を指定します。

```
   // Enable object lock on an existing bucket.
   public void enableObjectLockOnBucket(String bucketName) {
       try {
           VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
               .status(BucketVersioningStatus.ENABLED)
               .build();

           PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
               .bucket(bucketName)
               .versioningConfiguration(versioningConfiguration)
               .build();

           // Enable versioning on the bucket.
           getClient().putBucketVersioning(putBucketVersioningRequest);
           PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
               .bucket(bucketName)
               .objectLockConfiguration(ObjectLockConfiguration.builder()
                   .objectLockEnabled(ObjectLockEnabled.ENABLED)
                   .build())
               .build();

           getClient().putObjectLockConfiguration(request);
           System.out.println("Successfully enabled object lock on
"+bucketName);

       } catch (S3Exception ex) {
           System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
       }
   }
```

バケットのデフォルトの保存期間を設定します。

```
   // Set or modify a retention period on an S3 bucket.
   public void modifyBucketDefaultRetention(String bucketName) {
       VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
```

```
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();


getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName
+".");
    }
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「PutObjectLockConfiguration」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

バケットのオブジェクトロック設定を指定します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import {
  PutObjectLockConfigurationCommand,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 */
export const main = async (client, bucketName) => {
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
 bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
    },
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // Token: "OPTIONAL_TOKEN",
  });

  try {
    const response = await client.send(command);
    console.log(
      `Object Lock Configuration updated: ${response.$metadata.httpStatusCode}`,
    );
```

```
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME");
}
```

バケットのデフォルトの保存期間を設定します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import {
  PutObjectLockConfigurationCommand,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 */
export const main = async (client, bucketName) => {
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
 bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
      Rule: {
        DefaultRetention: {
          Mode: "GOVERNANCE",
          Years: 3,
        },
      },
    },
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // Token: "OPTIONAL_TOKEN",
```

```
  });

  try {
    const response = await client.send(command);
    console.log(
      `Default Object Lock Configuration updated: ${response.
$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME");
}
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「PutObjectLockConfiguration」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutObjectRetention** を使用する

以下のコード例は、PutObjectRetention の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Amazon S3 オブジェクトをロックする

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
```

```
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「PutObjectRetention」を参照してください。

CLI

AWS CLI

オブジェクトのオブジェクト保持設定を設定するには

次の put-object-retention 例では、指定されたオブジェクトのオブジェクト保持設定を 2025-01-01 まで設定します。

```
aws s3api put-object-retention \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf \
    --retention '{ "Mode": "GOVERNANCE", "RetainUntilDate":
"2025-01-01T00:00:00" }'
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「PutObjectRetention」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「PutObjectRetention」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   // Set or modify a retention period on an object in an S3 bucket.
   public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
       // Calculate the instant one day from now.
       Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

       // Convert the Instant to a ZonedDateTime object with a specific time
zone.
       ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

       // Define a formatter for human-readable output.
       DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

       // Format the ZonedDateTime object to a human-readable date string.
       String humanReadableDate = formatter.format(zonedDateTime);

       // Print the formatted date string.
       System.out.println("Formatted Date: " + humanReadableDate);
       ObjectLockRetention retention = ObjectLockRetention.builder()
           .mode(ObjectLockRetentionMode.GOVERNANCE)
           .retainUntilDate(futureInstant)
           .build();

       PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
           .bucket(bucketName)
           .key(objectKey)
           .retention(retention)
           .build();
```

```
        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
 until "+ humanReadableDate +".");
    }
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「PutObjectRetention」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import { PutObjectRetentionCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: objectKey,
    BypassGovernanceRetention: false,
    // ChecksumAlgorithm: "ALGORITHM",
    // ContentMD5: "MD5_HASH",
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    Retention: {
      Mode: "GOVERNANCE", // or "COMPLIANCE"
      RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
```

```
    },
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const response = await client.send(command);
    console.log(
      `Object Retention settings updated: ${response.$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME", "OBJECT_KEY");
}
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「PutObjectRetention」を参照してください。

PowerShell

Tools for PowerShell

例 1: このコマンドは、指定した S3 バケット内の「testfile.txt」オブジェクトの期限日「2019 年 12 月 31 日 00:00:00」までガバナンス保持モードを有効にします。

```
Write-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「PutObjectRetention」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **RestoreObject** を使用する

以下のコード例は、RestoreObject の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to restore an archived object in an Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class RestoreArchivedObject
{
    public static void Main()
    {
        string bucketName = "doc-example-bucket";
        string objectKey = "archived-object.txt";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        IAmazonS3 client = new AmazonS3Client(bucketRegion);
        RestoreObjectAsync(client, bucketName, objectKey).Wait();
    }

    /// <summary>
    /// This method restores an archived object from an Amazon S3 bucket.
    /// </summary>
```

```
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// RestoreObjectAsync.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object was located before it was archived.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object to restore.</param>
        public static async Task RestoreObjectAsync(IAmazonS3 client, string
bucketName, string objectKey)
        {
            try
            {
                var restoreRequest = new RestoreObjectRequest
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Days = 2,
                };
                RestoreObjectResponse response = await
client.RestoreObjectAsync(restoreRequest);

                // Check the status of the restoration.
                await CheckRestorationStatusAsync(client, bucketName, objectKey);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Error: {amazonS3Exception.Message}");
            }
        }

        /// <summary>
        /// This method retrieves the status of the object's restoration.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// GetObjectMetadataAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
        /// S3 bucket which contains the archived object.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object you want to restore.</param>
        public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
        {
```

```
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
            };

            GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

            var restStatus = response.RestoreInProgress ? "in-progress" :
"finished or failed";
            Console.WriteLine($"Restoration status: {restStatus}");
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「RestoreObject」を参照してください。

CLI

AWS CLI

オブジェクトの復元リクエストを作成するには

次の `restore-object` の例では、指定された Amazon S3 Glacier オブジェクトをバケット `my-glacier-bucket` に 10 日間復元します。

```
aws s3api restore-object \
    --bucket my-glacier-bucket \
    --key doc1.rtf \
    --restore-request Days=10
```

このコマンドでは何も出力されません。

- API の詳細については、AWS CLI コマンドリファレンスの「RestoreObject」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 *  For more information about restoring an object, see "Restoring an archived
 object" at
 *  https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 *  Before running this Java V2 code example, set up your development
 environment, including your credentials.
 *
 *  For more information, see the following documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <expectedBucketOwner>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name of an object with a Storage class
 value of Glacier.\s
```

```
                    expectedBucketOwner - The account that owns the bucket (you
can obtain this value from the AWS Management Console).\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }

    public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                    .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                    .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                    .expectedBucketOwner(expectedBucketOwner)
                    .bucket(bucketName)
                    .key(keyName)
                    .restoreRequest(restoreRequest)
                    .build();

            s3.restoreObject(objectRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「RestoreObject」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で `SelectObjectContent` を使用する

以下のコード例は、`SelectObjectContent` の使用方法を示しています。

CLI

AWS CLI

SQL ステートメントに基づいて Amazon S3 オブジェクトの内容をフィルタリングするには

次の `select-object-content` 例では、指定された SQL ステートメントで `my-data-file.csv` オブジェクトをフィルタリングし、出力をファイルに送信します。

```
aws s3api select-object-content \
    --bucket my-bucket \
    --key my-data-file.csv \
    --expression "select * from s3object limit 100" \
    --expression-type 'SQL' \
    --input-serialization '{"CSV": {}, "CompressionType": "NONE"}' \
    --output-serialization '{"CSV": {}}' "output.csv"
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「SelectObjectContent」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

次の例は、JSON オブジェクトを使用したクエリを示しています。完全な例では、CSV オブジェクトの使用も示しています。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import
 software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
```

```
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
 LoggerFactory.getLogger(SelectObjectContentExample.class);
    static final String BUCKET_NAME = "select-object-content-" +
 UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/
countries/master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
 SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where
 c.area < 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
                .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
                .compressionType(CompressionType.NONE)
                .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
                .json(JSONOutput.builder().recordDelimiter(null).build())
```

```
                .build();

        // Build the SelectObjectContentRequest.
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
                .bucket(BUCKET_NAME)
                .key(FILE_JSON)
                .expression(queryExpression)
                .expressionType(ExpressionType.SQL)
                .inputSerialization(inputSerialization)
                .outputSerialization(outputSerialization)
                .build();

        EventStreamInfo eventStreamInfo = new EventStreamInfo();
        // Call the selectObjectContent method with the request and a response
handler.
        // Supply an EventStreamInfo object to the response handler to gather
records and information from the response.
        s3AsyncClient.selectObjectContent(select,
buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record
->
                record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times",
fileType, eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler
buildResponseHandler(EventStreamInfo eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
SelectObjectContentResponseHandler.Visitor.builder()
                .onRecords(r -> {
                    logger.info("Record event received.");
                    eventStreamInfo.addRecord(r.payload().asUtf8String());
```

```
                    eventStreamInfo.incrementOnRecordsCalled();
                })
                .onCont(ce -> {
                    logger.info("Continuation event received.");
                    eventStreamInfo.incrementContinuationEvents();
                })
                .onProgress(pe -> {
                    Progress progress = pe.details();
                    logger.info("Progress event received:\n bytesScanned:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                            progress.bytesScanned(),
                            progress.bytesProcessed(),
                            progress.bytesReturned());
                })
                .onEnd(ee -> logger.info("End event received."))
                .onStats(se -> {
                    logger.info("Stats event received.");
                    eventStreamInfo.addStats(se.details());
                })
                .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
 processes the stream.
        return SelectObjectContentResponseHandler.builder()
                .subscriber(visitor).build();
    }

    // The EventStreamInfo class is used to store information gathered while
 processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
        private Integer countContinuationEvents = 0;
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
        }

        void incrementContinuationEvents() {
            countContinuationEvents++;
        }

        void addRecord(String record) {
```

```
            records.add(record);
        }

        void addStats(Stats stats) {
            this.stats = stats;
        }

        public List<String> getRecords() {
            return records;
        }

        public Integer getCountOnRecordsCalled() {
            return countOnRecordsCalled;
        }

        public Integer getCountContinuationEvents() {
            return countContinuationEvents;
        }

        public Stats getStats() {
            return stats;
        }
    }
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「SelectObjectContent」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **UploadPart** を使用する

以下のコード例は、UploadPart の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- マルチパートアップロードの実行
- チェックサムの使用

CLI

AWS CLI

次のコマンドは、create-multipart-upload コマンドで開始されたマルチパートアップロードの最初の部分をアップロードします。

```
aws s3api upload-part --bucket my-bucket --key 'multipart/01' --part-number 1 --
body part01 --upload-id
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
```

body オプションは、アップロードするローカルファイルの名前またはパスを指定します (file://プレフィックスは使用しないでください)。各パートの最小サイズは 5 MB です。アップロード ID は create-multipart-upload によって返され、list-multipart-uploads を使用して取得することもできます。バケットとキーは、マルチパートアップロードの作成時に指定されます。

出力:

```
{
    "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

後で使用できるように、各パートの ETag 値を保存します。これらはマルチパートアップロードを完了するために必要です。

- API の詳細については、AWS CLI コマンドリファレンスの「UploadPart」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        let upload_part_res = client
```

```
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;
        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );

   let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「UploadPart」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用した Amazon S3 のシナリオ

次のコード例は、AWS SDK を使用して Amazon S3 で一般的なシナリオを実装する方法を示しています。これらのシナリオは、Amazon S3 内で複数の関数を呼び出すことによって特定のタスクを実行する方法を示しています。それぞれのシナリオには、GitHub へのリンクがあり、コードを設定および実行する方法についての説明が記載されています。

例

- AWS SDK を使用して Amazon S3 の署名付き URL を作成する
- AWS SDK を使用して Amazon S3 オブジェクトを一覧表示するウェブページ
- AWS SDK を使用して Amazon S3 への不完全なマルチパートアップロードを削除する

- Amazon Simple Storage Service (Amazon S3) バケットからすべてのオブジェクトを、ローカルディレクトリにダウンロードする
- AWS SDK を使用してマルチリージョンアクセスポイントから Amazon S3 オブジェクトを取得する
- AWS SDK を使用して、Amazon S3 バケットから If-Modified-Since を指定してオブジェクトを取得する
- AWS SDK を使用して Amazon S3 バケットとオブジェクトの使用を開始する
- AWS SDK を使用して Amazon S3 オブジェクトの暗号化の使用を開始する
- AWS SDK を使用して Amazon S3 オブジェクトのタグの使用を開始する
- AWS SDK を使用して Amazon S3 オブジェクトのリーガルホールド設定を取得する
- AWS SDK を使用して Amazon S3 オブジェクトロック機能を操作する
- AWS SDK を使用して Amazon S3バケットのアクセスコントロールリスト (ACL) を管理する
- AWS SDK を使用して、バージョン管理されている Amazon S3 オブジェクトを Lambda 関数でバッチで管理する
- AWS SDK を使用して Amazon S3 の URI を解析する
- AWS SDK を使用して Amazon S3 オブジェクトのマルチパートコピーを実行する
- AWS SDK を使用して Amazon S3 オブジェクトのマルチパートアップロードを実行する
- AWS SDK を使用して Amazon S3 オブジェクトのアップロードまたはダウンロードを追跡する
- AWS SDK による単体テストと統合テストのアプローチ例
- ローカルディレクトリを Amazon Simple Storage Service (Amazon S3) バケットに、再帰的にアップロードする
- AWS SDK を使用して Amazon S3 との間で大きなファイルをアップロードまたはダウンロードする
- サイズが不明なストリームを AWS SDK を使用して Amazon S3 オブジェクトにアップロードする
- チェックサムを使用した AWS SDK での Amazon S3 オブジェクトの操作
- AWS SDK を使用して、Amazon S3 のバージョン管理されたオブジェクトを操作する

AWS SDK を使用して Amazon S3 の署名付き URL を作成する

次のコード例は、Amazon S3 の署名付き URL を作成し、オブジェクトをアップロードする方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

Amazon S3 アクションを期間限定で実行できる署名付き URL を生成します。

```
    using System;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;

    public class GenPresignedUrl
    {
        public static void Main()
        {
            const string bucketName = "doc-example-bucket";
            const string objectKey = "sample.txt";

            // Specify how long the presigned URL lasts, in hours
            const double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
 KMS, you must specify Signature Version 4.
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
```

```
            string urlString = GeneratePresignedURL(s3Client, bucketName,
objectKey, timeoutDuration);
            Console.WriteLine($"The generated URL is: {urlString}.");
        }

        /// <summary>
        /// Generate a presigned URL that can be used to access the file named
        /// in the objectKey parameter for the amount of time specified in the
        /// duration parameter.
        /// </summary>
        /// <param name="client">An initialized S3 client object used to call
        /// the GetPresignedUrl method.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// object for which to create the presigned URL.</param>
        /// <param name="objectKey">The name of the object to access with the
        /// presigned URL.</param>
        /// <param name="duration">The length of time for which the presigned
        /// URL will be valid.</param>
        /// <returns>A string representing the generated presigned URL.</returns>
        public static string GeneratePresignedURL(IAmazonS3 client, string
bucketName, string objectKey, double duration)
        {
            string urlString = string.Empty;
            try
            {
                var request = new GetPreSignedUrlRequest()
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration),
                };
                urlString = client.GetPreSignedURL(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:'{ex.Message}'");
            }

            return urlString;
        }
    }
```

署名済み URL を生成し、その URL を使用してアップロードを実行します。

```
    using System;
    using System.IO;
    using System.Net.Http;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket using a presigned URL. The code first
    /// creates a presigned URL and then uses it to upload an object to an
    /// Amazon S3 bucket using that URL.
    /// </summary>
    public class UploadUsingPresignedURL
    {
        private static HttpClient httpClient = new HttpClient();

        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "samplefile.txt";
            string filePath = $"source\\{keyName}";

            // Specify how long the signed URL will be valid in hours.
            double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
 KMS, you must specify Signature Version 4.
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);
```

```
            var url = GeneratePreSignedURL(client, bucketName, keyName,
timeoutDuration);
            var success = await UploadObject(filePath, url);

            if (success)
            {
                Console.WriteLine("Upload succeeded.");
            }
            else
            {
                Console.WriteLine("Upload failed.");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket using the presigned URL
passed in
        /// the url parameter.
        /// </summary>
        /// <param name="filePath">The path (including file name) to the local
        /// file you want to upload.</param>
        /// <param name="url">The presigned URL that will be used to upload the
        /// file to the Amazon S3 bucket.</param>
        /// <returns>A Boolean value indicating the success or failure of the
        /// operation, based on the HttpWebResponse.</returns>
        public static async Task<bool> UploadObject(string filePath, string url)
        {
            using var streamContent = new StreamContent(
                new FileStream(filePath, FileMode.Open, FileAccess.Read));

            var response = await httpClient.PutAsync(url, streamContent);
            return response.IsSuccessStatusCode;
        }

        /// <summary>
        /// Generates a presigned URL which will be used to upload an object to
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// GetPreSignedURL.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
```

```
        /// presigned URL will point.</param>
        /// <param name="objectKey">The name of the file that will be uploaded.</
param>
        /// <param name="duration">How long (in hours) the presigned URL will
        /// be valid.</param>
        /// <returns>The generated URL.</returns>
        public static string GeneratePreSignedURL(
            IAmazonS3 client,
            string bucketName,
            string objectKey,
            double duration)
        {
            var request = new GetPreSignedUrlRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Verb = HttpVerb.PUT,
                Expires = DateTime.UtcNow.AddHours(duration),
            };

            string url = client.GetPreSignedURL(request);
            return url;
        }
    }
```

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトをダウンロードするための署名付き URL を生成します。

```
//! Routine which demonstrates creating a pre-signed URL to download an object
 from an
//! Amazon Simple Storage Service (Amazon S3) bucket.
```

```
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param expirationSeconds: Expiration in seconds for pre-signed URL.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String
 &bucketName,
                                                       const Aws::String &key,
                                                       uint64_t expirationSeconds,
                                                       const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_GET,
                                       expirationSeconds);
}
```

libcurl を使用してダウンロードします。

```
static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to get an object from a bucket.
  \param resultString: A string to hold the result.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                 Aws::String &resultString) {
    CURL *curl = curl_easy_init();
    CURLcode result;
```

```
    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();

    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
 std::endl;
        return false;
    }

    return true;
}
```

オブジェクトをアップロードするための署名付き URL を生成します。

```
//! Routine which demonstrates creating a pre-signed URL to upload an object to
 an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String
 &bucketName,
                                                      const Aws::String &key,
                                                      uint64_t expirationSeconds,
                                                      const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_PUT,
                                       expirationSeconds);
}
```

libcurl を使用してアップロードします。

```
static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}
```

```
//! Utility routine to test putObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to put an object in a bucket.
  \param data: Body of the putObject request.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                 const Aws::String &data) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);
```

```
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    std::string outString = outWriteString.str();
    if (outString.empty()) {
        std::cout << "Successfully put object." << std::endl;
        return true;
    } else {
        std::cout << "A server error was encountered, output:\n" << outString
                  << std::endl;
        return false;
    }
}
```

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

S3 署名アクションをラップする関数を作成します。

```
// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
 actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
 S3.
// Presigned requests contain temporary credentials and can be made from any HTTP
 client.
type Presigner struct {
 PresignClient *s3.PresignClient
}



// GetObject makes a presigned request that can be used to get an object from a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
 bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignGetObject(context.TODO(),
 &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
```

```
 return request, err
}



// PutObject makes a presigned request that can be used to put an object in a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
 bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignPutObject(context.TODO(),
 &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// DeleteObject makes a presigned request that can be used to delete an object
 from a bucket.
func (presigner Presigner) DeleteObject(bucketName string, objectKey string)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignDeleteObject(context.TODO(),
 &s3.DeleteObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to delete object %v. Here's why:
 %v\n", objectKey, err)
 }
 return request, err
}
```

署名付き URL を生成して使用し、S3 オブジェクトをアップロード、ダウンロード、削除するインタラクティブな例を実行します。

```
// RunPresigningScenario is an interactive example that shows you how to get
 presigned
// HTTP requests that you can use to move data into and out of Amazon Simple
 Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and
 can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
// 2. Use the net/http package to use the presigned request to upload a local
 file to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the
 object to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
 mocked
// during testing.
func RunPresigningScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner, httpRequester IHttpRequester) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
```

```
log.Println("Welcome to the Amazon S3 presigning demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}
presignClient := s3.NewPresignClient(s3Client)
presigner := actions.Presigner{PresignClient: presignClient}

bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
 "you own or one you want to create:", demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(bucketName)
if err != nil {
 panic(err)
}
if !bucketExists {
 err = bucketBasics.CreateBucket(bucketName, sdkConfig.Region)
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
 demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
 panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedPutRequest.Method,
 presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
```

```
 panic(err)
}
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
 uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
 presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
 uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Let's presign a request to delete the object.")
questioner.Ask("Press Enter when you're ready.")
presignedDelRequest, err := presigner.DeleteObject(bucketName, uploadKey)
if err != nil {
```

```
  panic(err)
 }
 log.Printf("Got a presigned %v request to URL:\n\t%v\n",
 presignedDelRequest.Method,
  presignedDelRequest.URL)
 log.Println("Using net/http to send the request...")
 delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
 if err != nil {
  panic(err)
 }
 log.Printf("%v object %v with presigned URL returned %v.\n",
 presignedDelRequest.Method,
  uploadKey, delResponse.StatusCode)
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

この例で HTTP リクエストを行うために使用する HTTP リクエストラッパーを定義します。

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
 during
// unit testing.
type IHttpRequester interface {
 Get(url string) (resp *http.Response, err error)
 Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
 error)
 Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
 scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
 return http.Get(url)
}
func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
 (resp *http.Response, err error) {
```

```
 putRequest, err := http.NewRequest("PUT", url, body)
 if err != nil {
  return nil, err
 }
 putRequest.ContentLength = contentLength
 return http.DefaultClient.Do(putRequest)
}
func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error)
 {
 delRequest, err := http.NewRequest("DELETE", url, nil)
 if err != nil {
  return nil, err
 }
 return http.DefaultClient.Do(delRequest)
}
```

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトの署名付き URL を生成して、オブジェクトをダウンロードします (GET リクエスト)。

インポート。

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

URL を生成する。

```
   /* Create a pre-signed URL to download an object in a subsequent GET request.
*/
   public String createPresignedGetUrl(String bucketName, String keyName) {
       try (S3Presigner presigner = S3Presigner.create()) {

           GetObjectRequest objectRequest = GetObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .build();

           GetObjectPresignRequest presignRequest =
GetObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))  // The URL will
expire in 10 minutes.
                   .getObjectRequest(objectRequest)
                   .build();
```

```
            PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
            logger.info("Presigned URL: [{}]",
presignedRequest.url().toString());
            logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
```

次の 3 つの方法のいずれかを使用して、オブジェクトをダウンロードします。

JDK HttpURLConnection (v1.1 以降) クラスを使用してダウンロードを行います。

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
    public byte[] useHttpUrlConnectionToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setRequestMethod("GET");
            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                IoUtils.copy(content, byteArrayOutputStream);
            }
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

JDK HttpClient (v11 以降) クラスを使用してダウンロードを行います。

```
    /* Use the JDK HttpClient (since v11) class to do the download. */
```

```
   public byte[] useHttpClientToGet(String presignedUrlString) {
       ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

       HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
       HttpClient httpClient = HttpClient.newHttpClient();
       try {
           URL presignedUrl = new URL(presignedUrlString);
           HttpResponse<InputStream> response = httpClient.send(requestBuilder
                           .uri(presignedUrl.toURI())
                           .GET()
                           .build(),
                   HttpResponse.BodyHandlers.ofInputStream());

           IoUtils.copy(response.body(), byteArrayOutputStream);

           logger.info("HTTP response code is " + response.statusCode());

       } catch (URISyntaxException | InterruptedException | IOException e) {
           logger.error(e.getMessage(), e);
       }
       return byteArrayOutputStream.toByteArray();
   }
```

AWS SDK for Java SdkHttpClient クラスを使用してダウンロードを行います。

```
   /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
   public byte[] useSdkHttpClientToPut(String presignedUrlString) {

       ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
       try {
           URL presignedUrl = new URL(presignedUrlString);
           SdkHttpRequest request = SdkHttpRequest.builder()
                   .method(SdkHttpMethod.GET)
                   .uri(presignedUrl.toURI())
                   .build();

           HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                   .request(request)
                   .build();
```

```
            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                response.responseBody().ifPresentOrElse(
                        abortableInputStream -> {
                            try {
                                IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                            } catch (IOException e) {
                                throw new RuntimeException(e);
                            }
                        },
                        () -> logger.error("No response body."));

                logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

アップロード用の署名付き URL を生成して、ファイルをアップロードします (PUT リクエスト)。

インポート。

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
```

```
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

URL を生成する。

```
   /* Create a presigned URL to use in a subsequent PUT request */
   public String createPresignedUrl(String bucketName, String keyName,
Map<String, String> metadata) {
       try (S3Presigner presigner = S3Presigner.create()) {

           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .metadata(metadata)
                   .build();

           PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))  // The URL
expires in 10 minutes.
                   .putObjectRequest(objectRequest)
                   .build();
```

```
            PresignedPutObjectRequest presignedRequest =
 presigner.presignPutObject(presignRequest);
            String myURL = presignedRequest.url().toString();
            logger.info("Presigned URL to upload a file to: [{}]", myURL);
            logger.info("HTTP method: [{}]",
 presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
```

次の 3 つの方法のいずれかを使用して、オブジェクトをアップロードします。

JDK HttpURLConnection (v1.1 以降) クラスを使用してアップロードを行います。

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
    public void useHttpUrlConnectionToPut(String presignedUrlString, File
 fileToPut, Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
 presignedUrl.openConnection();
            connection.setDoOutput(true);
            metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-
meta-" + k, v));
            connection.setRequestMethod("PUT");
            OutputStream out = connection.getOutputStream();

            try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
                 FileChannel inChannel = file.getChannel()) {
                ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is
 8k

                while (inChannel.read(buffer) > 0) {
                    buffer.flip();
                    for (int i = 0; i < buffer.limit(); i++) {
                        out.write(buffer.get());
                    }
                    buffer.clear();
                }
```

```
            } catch (IOException e) {
                logger.error(e.getMessage(), e);
            }

            out.close();
            connection.getResponseCode();
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

JDK HttpClient (v11 以降) クラスを使用してアップロードを行います。

```
    /* Use the JDK HttpClient (since v11) class to do the upload. */
    public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            final HttpResponse<Void> response = httpClient.send(requestBuilder
                            .uri(new URL(presignedUrlString).toURI())

.PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
                            .build(),
                    HttpResponse.BodyHandlers.discarding());

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

AWS for Java V2 SdkHttpClient クラスを使用してアップロードを行います。

```
   /* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
   public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
       logger.info("Begin [{}] upload", fileToPut.toString());

       try {
           URL presignedUrl = new URL(presignedUrlString);

           SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                   .method(SdkHttpMethod.PUT)
                   .uri(presignedUrl.toURI());
           // Add headers
           metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" +
k, v));
           // Finish building the request.
           SdkHttpRequest request = requestBuilder.build();

           HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                   .request(request)
                   .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
                   .build();

           try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
               HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
               logger.info("Response code: {}",
response.httpResponse().statusCode());
           }
       } catch (URISyntaxException | IOException e) {
           logger.error(e.getMessage(), e);
       }
   }
```

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

署名付き URL を作成して、オブジェクトをバケットにアップロードします。

```
import https from "https";
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" }),
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};
```

```
function put(url, data) {
  return new Promise((resolve, reject) => {
    const req = https.request(
      url,
      { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
      (res) => {
        let responseBody = "";
        res.on("data", (chunk) => {
          responseBody += chunk;
        });
        res.on("end", () => {
          resolve(responseBody);
        });
      },
    );
    req.on("error", (err) => {
      reject(err);
    });
    req.write(data);
    req.end();
  });
}

export const main = async () => {
  const REGION = "us-east-1";
  const BUCKET = "example_bucket";
  const KEY = "example_file.txt";

  // There are two ways to generate a presigned URL.
  // 1. Use createPresignedUrl without the S3 client.
  // 2. Use getSignedUrl in conjunction with the S3 client and GetObjectCommand.
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    const clientUrl = await createPresignedUrlWithClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });
```

```
    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");

    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
  } catch (err) {
    console.error(err);
  }
};
```

署名付き URL を作成して、オブジェクトをバケットからダウンロードします。

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(new HttpRequest(url));
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
```

```
  const command = new GetObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

export const main = async () => {
  const REGION = "us-east-1";
  const BUCKET = "example_bucket";
  const KEY = "example_file.jpg";

  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    const clientUrl = await createPresignedUrlWithClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    console.log("Presigned URL without client");
    console.log(noClientUrl);
    console.log("\n");

    console.log("Presigned URL with client");
    console.log(clientUrl);
  } catch (err) {
    console.error(err);
  }
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

GetObject の署名済みリクエストを作成し、その URL を使用してオブジェクトをダウンロードします。

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET
 request to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()

    return objectContents
}
```

詳細オプションを使用して GetObject 署名済みリクエストを作成します。

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
```

```
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be
 used 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

PutObject の署名済みリクエストを作成し、それを使用してオブジェクトをアップロードします。

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
 HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
```

```
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }.put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- 詳細については、「AWS SDK for Kotlin デベロッパーガイド」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
```

```
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the
 S3 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in
 the given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
 next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
$runner->run();
```

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

S3 アクションを期間限定で実行できる署名付き URL を生成します。Requests パッケージを使用して、URL でリクエストを行います。

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method, Params=method_parameters,
 ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method
```

```
        )
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key",
        help="For a GET operation, the key of the object in Amazon S3. For a "
        "PUT operation, the name of a file to upload.",
    )
    parser.add_argument("action", choices=("get", "put"), help="The action to
 perform.")
    args = parser.parse_args()

    s3_client = boto3.client("s3")
    client_action = "get_object" if args.action == "get" else "put_object"
    url = generate_presigned_url(
        s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
    )

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == "get":
        response = requests.get(url)
    elif args.action == "put":
        print("Putting data to the URL.")
        try:
            with open(args.key, "r") as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(
                f"Couldn't find {args.key}. For a PUT operation, the key must be
 the "
                f"name of a file that exists on your computer."
```

```
            )

    if response is not None:
        print("Got response:")
        print(f"Status: {response.status_code}")
        print(response.text)

    print("-" * 88)


if __name__ == "__main__":
    usage_demo()
```

署名付き POST リクエストを生成して、ファイルをアップロードします。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an
 AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.
        :param expires_in: The number of seconds the presigned POST is valid.
        :return: A dictionary that contains the URL and form fields that contain
                 required access data.
        """
        try:
            response = self.bucket.meta.client.generate_presigned_post(
```

```
                Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
            )
            logger.info("Got presigned POST URL: %s", response["url"])
        except ClientError:
            logger.exception(
                "Couldn't get a presigned POST URL for bucket '%s' and object
 '%s'",
                self.bucket.name,
                object_key,
            )
            raise
        return response
```

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require "aws-sdk-s3"
require "net/http"

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's
 why: #{e.message}"
```

```
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request("PUT", presigned_url.request_uri, object_content,
 "content_type" => "")
  end

  case response
  when Net::HTTPSuccess
    puts "Content uploaded!"
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

S3 オブジェクトの GET および PUT への事前署名リクエストを作成します。

```
async fn get_object(
    client: &Client,
```

```
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);
    let presigned_request = client
        .get_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;

    println!("Object URI: {}", presigned_request.uri());

    Ok(())
}

async fn put_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);

    let presigned_request = client
        .put_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;

    println!("Object URI: {}", presigned_request.uri());

    Ok(())
}
```

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 オブジェクトを一覧表示するウェブページ

次のコード例は、ウェブページに Amazon S3 オブジェクトを一覧表示する方法を示しています。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

以下のコードは、AWS SDK を呼び出す、関連する React コンポーネントです。このコンポーネントを含むアプリケーションの実行可能なバージョンは、前述の GitHub リンクにあります。

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
  const [objects, setObjects] = useState<
    Required<ListObjectsCommandOutput>["Contents"]
  >([]);

  useEffect(() => {
    const client = new S3Client({
      region: "us-east-1",
      // Unless you have a public bucket, you'll need access to a private bucket.
      // One way to do this is to create an Amazon Cognito identity pool, attach
 a role to the pool,
      // and grant the role access to the 's3:GetObject' action.
      //
```

```
      // You'll also need to configure the CORS settings on the bucket to allow
 traffic from
      // this example site. Here's an example configuration that allows all
 origins. Don't
      // do this in production.
      //[
      //  {
      //    "AllowedHeaders": ["*"],
      //    "AllowedMethods": ["GET"],
      //    "AllowedOrigins": ["*"],
      //    "ExposeHeaders": [],
      //  },
      //]
      //
      credentials: fromCognitoIdentityPool({
        clientConfig: { region: "us-east-1" },
        identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
      }),
    });
    const command = new ListObjectsCommand({ Bucket: "bucket-name" });
    client.send(command).then(({ Contents }) => setObjects(Contents || []));
  }, []);

  return (
    <div className="App">
      {objects.map((o) => (
        <div key={o.ETag}>{o.Key}</div>
      ))}
    </div>
  );
}

export default App;
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「ListObjects」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 への不完全なマルチパートアップロードを削除する

次のコード例は、不完全な Amazon S3 マルチパートアップロードを削除または停止する方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

何らかの理由で進行中または不完全なマルチパートアップロードを停止するには、次の例に示すように、リストアップロードを取得してから削除します。

```
    public static void abortIncompleteMultipartUploadsFromList() {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
```

```
                logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
            }
        }
    }
```

日付の前後に開始された不完全なマルチパートアップロードを削除するには、次の例に示すように、特定の時点に基づいてマルチパートアップロードを選択的に削除します。

```
    static void abortIncompleteMultipartUploadsOlderThan(Instant pointInTime) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(upload.key())
                        .expectedBucketOwner(accountId)
                        .uploadId(upload.uploadId())
                        .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }
```

マルチパートアップロードの開始後にアップロード ID にアクセスできる場合は、その ID を使用して進行中のアップロードを削除できます。

```
    static void abortMultipartUploadUsingUploadId() {
        String uploadId = startUploadReturningUploadId();
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
                .uploadId(uploadId)
                .bucket(bucketName)
                .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
 uploadId, bucketName);
        }
    }
```

特定の日数だけ古い不完全なマルチパートアップロードを一貫して削除するには、バケットのバケットライフサイクル設定を設定します。次の例は、7 日以上経過した不完全なアップロードを削除するルールを作成する方法を示しています。

```
    static void abortMultipartUploadsUsingLifecycleConfig() {
        Collection<LifecycleRule> lifeCycleRules =
 List.of(LifecycleRule.builder()
                .abortIncompleteMultipartUpload(b -> b.
                        daysAfterInitiation(7))
                .status("Enabled")
                .filter(SdkBuilder::build) // Filter element is required.
                .build());

        // If the action is successful, the service sends back an HTTP 200
 response with an empty HTTP body.
        PutBucketLifecycleConfigurationResponse response =
 s3Client.putBucketLifecycleConfiguration(b -> b
                .bucket(bucketName)
                .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
```

```
            logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
        } else {
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
        }
    }
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - AbortMultipartUpload
  - ListMultipartUploads
  - PutBucketLifecycleConfiguration

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon Simple Storage Service (Amazon S3) バケットからすべてのオブジェクトを、ローカルディレクトリにダウンロードする

次のコード例は、Amazon Simple Storage Service (Amazon S3) バケットからすべてのオブジェクトをローカルディレクトリにダウンロードする方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

S3TransferManager を使用して、同じ S3 バケットにあるすべての S3 オブジェクトをダウンロードします。完全なファイルとテストを表示します。

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「DownloadDirectory」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用してマルチリージョンアクセスポイントから Amazon S3 オブジェクトを取得する

次のコード例は、マルチリージョンアクセスポイントからオブジェクトを取得する方法を示しています。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

非対称 Sigv4 (Sigv4a) 署名アルゴリズムを使用するように S3 クライアントを設定します。

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric Sigv4 (Sigv4a)
signing algorithm.
            val sigV4AScheme = SigV4AsymmetricAuthScheme(CrtAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4AScheme)
            }
            return s3
        }
```

バケット名の代わりにマルチリージョンアクセスポイント ARN を使用してオブジェクトを取得します。

```
    suspend fun getObjectFromMrap(
        s3: S3Client,
        mrapArn: String,
        keyName: String,
    ): String? {
        val request = GetObjectRequest {
            bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
            key = keyName
```

```
        }

        var stringObj: String? = null
        s3.getObject(request) { resp ->
            stringObj = resp.body?.decodeToString()
            if (stringObj != null) {
                println("Successfully read $keyName from $mrapArn")
            }
        }
        return stringObj
    }
```

- 詳細については、「AWS SDK for Kotlin デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetObject」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して、Amazon S3 バケットから If-Modified-Since を指定してオブジェクトを取得する

次のコード例は、S3 バケット内のオブジェクトからデータを読み取る方法を示しています。ただし、そのバケットが前回の取得時刻以降に変更されていない場合に限ります。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
use aws_sdk_s3::{
    error::SdkError,
    operation::head_object::HeadObjectError,
```

```
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client, Error,
};
use tracing::{error, warn};

const KEY: &str = "key";
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();

    // Get a new UUID to use when creating a unique bucket name.
    let uuid = uuid::Uuid::new_v4();

    // Load the AWS configuration from the environment.
    let client = Client::new(&aws_config::load_from_env().await);

    // Generate a unique bucket name using the previously generated UUID.
    // Then create a new bucket with that name.
    let bucket_name = format!("if-modified-since-{uuid}");
    client
        .create_bucket()
        .bucket(bucket_name.clone())
        .send()
        .await?;

    // Create a new object in the bucket whose name is `KEY` and whose
    // contents are `BODY`.
    let put_object_output = client
        .put_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .body(ByteStream::from_static(BODY.as_bytes()))
        .send()
```

```
        .await;

    // If the `PutObject` succeeded, get the eTag string from it. Otherwise,
    // report an error and return an empty string.
    let e_tag_1 = match put_object_output {
        Ok(put_object) => put_object.e_tag.unwrap(),
        Err(err) => {
            error!("{err:?}");
            String::new()
        }
    };

    // Request the object's headers.
    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await;

    // If the `HeadObject` request succeeded, create a tuple containing the
    // values of the headers `last-modified` and `etag`. If the request
    // failed, return the error in a tuple instead.
    let (last_modified, e_tag_2) = match head_object_output {
        Ok(head_object) => (
            Ok(head_object.last_modified().cloned().unwrap()),
            head_object.e_tag.unwrap(),
        ),
        Err(err) => (Err(err), String::new()),
    };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and first GetObject had differing eTags"
    );

    println!("First value of last_modified: {last_modified:?}");
    println!("First tag: {}\n", e_tag_1);

    // Send a second `HeadObject` request. This time, the `if_modified_since`
    // option is specified, giving the `last_modified` value returned by the
    // first call to `HeadObject`.
    //
```

```
    // Since the object hasn't been changed, and there are no other objects in
    // the bucket, there should be no matching objects.

    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .if_modified_since(last_modified.unwrap())
        .send()
        .await;

    // If the `HeadObject` request succeeded, the result is a typle containing
    // the `last_modified` and `e_tag_1` properties. This is _not_ the expected
    // result.
    //
    // The _expected_ result of the second call to `HeadObject` is an
    // `SdkError::ServiceError` containing the HTTP error response. If that's
    // the case and the HTTP status is 304 (not modified), the output is a
    // tuple containing the values of the HTTP `last-modified` and `etag`
    // headers.
    //
    // If any other HTTP error occurred, the error is returned as an
    // `SdkError::ServiceError`.

    let (last_modified, e_tag_2): (Result<DateTime, SdkError<HeadObjectError>>,
String) =
        match head_object_output {
            Ok(head_object) => (
                Ok(head_object.last_modified().cloned().unwrap()),
                head_object.e_tag.unwrap(),
            ),
            Err(err) => match err {
                SdkError::ServiceError(err) => {
                    // Get the raw HTTP response. If its status is 304, the
                    // object has not changed. This is the expected code path.
                    let http = err.raw();
                    match http.status().as_u16() {
                        // If the HTTP status is 304: Not Modified, return a
                        // tuple containing the values of the HTTP
                        // `last-modified` and `etag` headers.
                        304 => (
                            Ok(DateTime::from_str(
                                http.headers().get("last-modified").unwrap(),
                                DateTimeFormat::HttpDate,
```

```
                            )
                            .unwrap()),
                            http.headers().get("etag").map(|t|
 t.into()).unwrap(),
                        ),
                        // Any other HTTP status code is returned as an
                        // `SdkError::ServiceError`.
                        _ => (Err(SdkError::ServiceError(err)), String::new()),
                    }
                }
                // Any other kind of error is returned in a tuple containing the
                // error and an empty string.
                _ => (Err(err), String::new()),
            },
        };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and second HeadObject had different eTags"
    );

    println!("Second value of last modified: {last_modified:?}");
    println!("Second tag: {}", e_tag_2);

    // Clean up by deleting the object and the bucket.
    client
        .delete_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await?;

    client
        .delete_bucket()
        .bucket(bucket_name.as_str())
        .send()
        .await?;

    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「GetObject」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 バケットとオブジェクトの使用を開始する

次のコード例は、以下を実行する方法を示しています。

- バケットを作成し、そこにファイルをアップロードします。
- バケットからオブジェクトをダウンロードします。
- バケット内のサブフォルダにオブジェクトをコピーします。
- バケット内のオブジェクトを一覧表示します。
- バケットオブジェクトとバケットを削除します。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
public class S3_Basics
{
    public static async Task Main()
    {
        // Create an Amazon S3 client object. The constructor uses the
        // default user installed on the system. To work with Amazon S3
        // features in a different AWS Region, pass the AWS Region as a
        // parameter to the client constructor.
        IAmazonS3 client = new AmazonS3Client();
        string bucketName = string.Empty;
```

```
            string filePath = string.Empty;
            string keyName = string.Empty;

            var sepBar = new string('-', Console.WindowWidth);

            Console.WriteLine(sepBar);
            Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
            Console.WriteLine("procedures. This application will:");
            Console.WriteLine("\n\t1. Create a bucket");
            Console.WriteLine("\n\t2. Upload an object to the new bucket");
            Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
 bucket");
            Console.WriteLine("\n\t4. List the items in the new bucket");
            Console.WriteLine("\n\t5. Delete all the items in the bucket");
            Console.WriteLine("\n\t6. Delete the bucket");
            Console.WriteLine(sepBar);

            // Create a bucket.
            Console.WriteLine($"\n{sepBar}");
            Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");
            Console.WriteLine(sepBar);

            Console.Write("Please enter a name for the new bucket: ");
            bucketName = Console.ReadLine();

            var success = await S3Bucket.CreateBucketAsync(client, bucketName);
            if (success)
            {
                Console.WriteLine($"Successfully created bucket: {bucketName}.
\n");
            }
            else
            {
                Console.WriteLine($"Could not create bucket: {bucketName}.\n");
            }

            Console.WriteLine(sepBar);
            Console.WriteLine("Upload a file to the new bucket.");
            Console.WriteLine(sepBar);

            // Get the local path and filename for the file to upload.
            while (string.IsNullOrEmpty(filePath))
            {
```

```
                Console.Write("Please enter the path and filename of the file to
upload: ");
                filePath = Console.ReadLine();

                // Confirm that the file exists on the local computer.
                if (!File.Exists(filePath))
                {
                    Console.WriteLine($"Couldn't find {filePath}. Try again.\n");
                    filePath = string.Empty;
                }
            }

            // Get the file name from the full path.
            keyName = Path.GetFileName(filePath);

            success = await S3Bucket.UploadFileAsync(client, bucketName, keyName,
filePath);

            if (success)
            {
                Console.WriteLine($"Successfully uploaded {keyName} from
{filePath} to {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Could not upload {keyName}.\n");
            }

            // Set the file path to an empty string to avoid overwriting the
            // file we just uploaded to the bucket.
            filePath = string.Empty;

            // Now get a new location where we can save the file.
            while (string.IsNullOrEmpty(filePath))
            {
                // First get the path to which the file will be downloaded.
                Console.Write("Please enter the path where the file will be
downloaded: ");
                filePath = Console.ReadLine();

                // Confirm that the file exists on the local computer.
                if (File.Exists($"{filePath}\\{keyName}"))
                {
```

```
                    Console.WriteLine($"Sorry, the file already exists in that
location.\n");
                    filePath = string.Empty;
                }
            }

            // Download an object from a bucket.
            success = await S3Bucket.DownloadObjectFromBucketAsync(client,
bucketName, keyName, filePath);

            if (success)
            {
                Console.WriteLine($"Successfully downloaded {keyName}.\n");
            }
            else
            {
                Console.WriteLine($"Sorry, could not download {keyName}.\n");
            }

            // Copy the object to a different folder in the bucket.
            string folderName = string.Empty;

            while (string.IsNullOrEmpty(folderName))
            {
                Console.Write("Please enter the name of the folder to copy your
object to: ");
                folderName = Console.ReadLine();
            }

            while (string.IsNullOrEmpty(keyName))
            {
                // Get the name to give to the object once uploaded.
                Console.Write("Enter the name of the object to copy: ");
                keyName = Console.ReadLine();
            }

            await S3Bucket.CopyObjectInBucketAsync(client, bucketName, keyName,
folderName);

            // List the objects in the bucket.
            await S3Bucket.ListBucketContentsAsync(client, bucketName);

            // Delete the contents of the bucket.
            await S3Bucket.DeleteBucketContentsAsync(client, bucketName);
```

```
            // Deleting the bucket too quickly after deleting its contents will
            // cause an error that the bucket isn't empty. So...
            Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
            _ = Console.ReadLine();

            // Delete the bucket.
            await S3Bucket.DeleteBucketAsync(client, bucketName);
        }
    }
```

- API の詳細については、『AWS SDK for .NET API リファレンス』の以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Bash

Bash スクリプトを使用した AWS CLI

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
#############################################################################
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
```

```
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function s3_getting_started() {
  {
    if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
      cd bucket-lifecycle-operations || exit

      source ./bucket_operations.sh
      cd ..
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the Amazon S3 getting started demo."
  echo_repeat "*" 88

  local bucket_name
  bucket_name=$(generate_random_name "doc-example-bucket")

  local region_code
  region_code=$(aws configure get region)

  if create_bucket -b "$bucket_name" -r "$region_code"; then
    echo "Created demo bucket named $bucket_name"
  else
    errecho "The bucket failed to create. This demo will exit."
    return 1
  fi

  local file_name
  while [ -z "$file_name" ]; do
    echo -n "Enter a file you want to upload to your bucket: "
    get_input
    file_name=$get_input_result

    if [ ! -f "$file_name" ]; then
      echo "Could not find file $file_name. Are you sure it exists?"
      file_name=""
    fi
  done
```

```
 local key
 key="$(basename "$file_name")"

 local result=0
 if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
   echo "Uploaded file $file_name into bucket $bucket_name with key $key."
 else
   result=1
 fi

 local destination_file
 destination_file="$file_name.download"
 if yes_no_input "Would you like to download $key to the file $destination_file?
(y/n) "; then
   if download_object_from_bucket "$bucket_name" "$destination_file" "$key";
then
     echo "Downloaded $key in the bucket $bucket_name to the file
$destination_file."
   else
     result=1
   fi
 fi

 if yes_no_input "Would you like to copy $key a new object key in your bucket?
(y/n) "; then
   local to_key
   to_key="demo/$key"
   if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
     echo "Copied $key in the bucket $bucket_name to the  $to_key."
   else
     result=1
   fi
 fi

 local bucket_items
 bucket_items=$(list_items_in_bucket "$bucket_name")

 # shellcheck disable=SC2181
 if [[ $? -ne 0 ]]; then
   result=1
 fi

 echo "Your bucket contains the following items."
 echo -e "Name\t\tSize"
```

```
  echo "$bucket_items"

  if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
 (y/n) "; then
    bucket_items=$(echo "$bucket_items" | cut -f 1)

    if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
      echo "The following items were deleted from the bucket $bucket_name"
      echo "$bucket_items"
    else
      result=1
    fi

    if delete_bucket "$bucket_name"; then
      echo "Deleted the bucket $bucket_name"
    else
      result=1
    fi
  fi

  return $result
}
```

このシナリオで使用される Amazon S3 関数。

```
##############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
```

```
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally
 unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi
```

```
  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}

################################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3
```

```
  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

################################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

```
##############################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}

##############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
```

```
#     And:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}

################################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
```

```
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}

################################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」で以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"
```

```
namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param client: An S3 client.
          \return bool: Function succeeded.
        */
        static bool
        deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        //! Delete an object in an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param key: The key for the object in the S3 bucket.
          \param client: An S3 client.
          \return bool: Function succeeded.
         */
        static bool
        deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
 &key,
                               Aws::S3::S3Client &client);
    }
}


//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
  \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
  \param saveFilePath: Path for saving a downloaded S3 object.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &uploadFilePath,
                                           const Aws::String &saveFilePath,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: "doc-example-bucket-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
```

```
    Aws::String bucketName = "doc-example-bucket-" +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(

Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                            clientConfig.region));
            request.WithCreateBucketConfiguration(createBucketConfiguration);
        }

        Aws::S3::Model::CreateBucketOutcome outcome =
client.CreateBucket(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: createBucket: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
            return false;
        } else {
            std::cout << "Created the bucket, '" << bucketName <<
                      "', in the region, '" << clientConfig.region << "'." <<
std::endl;
        }
    }

    // 2. Upload a local file to the bucket.
    Aws::String key = "key-for-test";
    {
        Aws::S3::Model::PutObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        std::shared_ptr<Aws::FStream> input_data =
                Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                              uploadFilePath,
                                              std::ios_base::in |
```

```
                                                std::ios_base::binary);

        if (!input_data->is_open()) {
            std::cerr << "Error: unable to open file, '" << uploadFilePath <<
"'."
                      << std::endl;
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        }

        request.SetBody(input_data);

        Aws::S3::Model::PutObjectOutcome outcome =
                client.PutObject(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: putObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
            AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        } else {
            std::cout << "Added the object with the key, '" << key
                      << "', to the bucket, '"
                      << bucketName << "'." << std::endl;
        }
    }

    // 3. Download the object to a local file.
    {
        Aws::S3::Model::GetObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        Aws::S3::Model::GetObjectOutcome outcome =
                client.GetObject(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: getObject: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
        } else {
            std::cout << "Downloaded the object with the key, '" << key
```

```
                    << "', in the bucket, '"
                    << bucketName << "'." << std::endl;

            Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
                    GetBody();
            Aws::OFStream outStream(saveFilePath,
                                    std::ios_base::out | std::ios_base::binary);
            if (!outStream.is_open()) {
                std::cout << "Error: unable to open file, '" << saveFilePath <<
"'."
                          << std::endl;
            } else {
                outStream << ioStream.rdbuf();
                std::cout << "Wrote the downloaded object to the file '"
                          << saveFilePath << "'." << std::endl;
            }
        }
    }

    // 4. Copy the object to a different "folder" in the bucket.
    Aws::String copiedToKey = "test-folder/" + key;
    {
        Aws::S3::Model::CopyObjectRequest request;
        request.WithBucket(bucketName)
                .WithKey(copiedToKey)
                .WithCopySource(bucketName + "/" + key);

        Aws::S3::Model::CopyObjectOutcome outcome =
                client.CopyObject(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error: copyObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
        } else {
            std::cout << "Copied the object with the key, '" << key
                      << "', to the key, '" << copiedToKey
                      << ", in the bucket, '" << bucketName << "'." << std::endl;
        }
    }

    // 5. List objects in the bucket.
    {
        Aws::S3::Model::ListObjectsV2Request request;
        request.WithBucket(bucketName);
```

```
        Aws::String continuationToken;
        Aws::Vector<Aws::S3::Model::Object> allObjects;

        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
                    request);

            if (!outcome.IsSuccess()) {
                std::cerr << "Error: ListObjects: " <<
                          outcome.GetError().GetMessage() << std::endl;
                break;
            } else {
                Aws::Vector<Aws::S3::Model::Object> objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(),
 objects.end());
                continuationToken = outcome.GetResult().GetContinuationToken();
            }
        } while (!continuationToken.empty());

        std::cout << allObjects.size() << " objects in the bucket, '" <<
 bucketName
                  << "':" << std::endl;

        for (Aws::S3::Model::Object &object: allObjects) {
            std::cout << "     '" << object.GetKey() << "'" << std::endl;
        }
    }

    // 6. Delete all objects in the bucket.
    // All objects in the bucket must be deleted before deleting the bucket.
    AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
    AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

    // 7. Delete the bucket.
    return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {
```

```
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Deleted the object with the key, '" << key
                  << "', from the bucket, '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client
 &client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
    }
    return outcome.IsSuccess();
}
```

- API の詳細については、『AWS SDK for C++ API リファレンス』の以下のトピックを参照してください。
  - CopyObject

- CreateBucket
- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

シナリオで使用されるバケットアクションとオブジェクトアクションをラップする構造体を定義します。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets() ([]types.Bucket, error) {
 result, err := basics.S3Client.ListBuckets(context.TODO(),
 &s3.ListBucketsInput{})
 var buckets []types.Bucket
 if err != nil {
```

```
  log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
 } else {
  buckets = result.Buckets
 }
 return buckets, err
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(bucketName string) (bool, error) {
 _, err := basics.S3Client.HeadBucket(context.TODO(), &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
    log.Printf("Either you don't have access to bucket %v or another error
 occurred. "+
     "Here's what happened: %v\n", bucketName, err)
   }
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(name string, region string) error {
 _, err := basics.S3Client.CreateBucket(context.TODO(), &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
```

```
  },
 })
 if err != nil {
  log.Printf("Couldn't create bucket %v in Region %v. Here's why: %v\n",
   name, region, err)
 }
 return err
}



// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(bucketName string, objectKey string,
 fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(context.TODO(), &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
    fileName, bucketName, objectKey, err)
  }
 }
 return err
}



// UploadLargeObject uses an upload manager to upload data to an object in a
 bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(bucketName string, objectKey string,
 largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
```

```
 })
 _, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }

 return err
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(bucketName string, objectKey string,
 fileName string) error {
 result, err := basics.S3Client.GetObject(context.TODO(), &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
 objectKey, err)
  return err
 }
 defer result.Body.Close()
 file, err := os.Create(fileName)
 if err != nil {
  log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
  return err
 }
 defer file.Close()
 body, err := io.ReadAll(result.Body)
 if err != nil {
  log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
 err)
 }
 _, err = file.Write(body)
 return err
}
```

```
// DownloadLargeObject uses a download manager to download an object from a
 bucket.
// The download manager gets the data in parts and writes them to a buffer until
 all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(bucketName string, objectKey
 string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
 {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(context.TODO(), buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}



// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(bucketName string, objectKey string,
 folderName string) error {
 _, err := basics.S3Client.CopyObject(context.TODO(), &s3.CopyObjectInput{
  Bucket:     aws.String(bucketName),
  CopySource: aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
  Key:        aws.String(fmt.Sprintf("%v/%v", folderName, objectKey)),
 })
 if err != nil {
  log.Printf("Couldn't copy object from %v:%v to %v:%v/%v. Here's why: %v\n",
   bucketName, objectKey, bucketName, folderName, objectKey, err)
 }
 return err
}
```

```
// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(sourceBucket string, destinationBucket
 string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(context.TODO(), &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't copy object from %v:%v to %v:%v. Here's why: %v\n",
   sourceBucket, objectKey, destinationBucket, objectKey, err)
 }
 return err
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(bucketName string) ([]types.Object, error)
 {
 result, err := basics.S3Client.ListObjectsV2(context.TODO(),
 &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 })
 var contents []types.Object
 if err != nil {
  log.Printf("Couldn't list objects in bucket %v. Here's why: %v\n", bucketName,
 err)
 } else {
  contents = result.Contents
 }
 return contents, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(bucketName string, objectKeys []string)
 error {
 var objectIds []types.ObjectIdentifier
 for _, key := range objectKeys {
  objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
 }
```

```
 output, err := basics.S3Client.DeleteObjects(context.TODO(),
 &s3.DeleteObjectsInput{
  Bucket: aws.String(bucketName),
  Delete: &types.Delete{Objects: objectIds},
 })
 if err != nil {
  log.Printf("Couldn't delete objects from bucket %v. Here's why: %v\n",
 bucketName, err)
 } else {
  log.Printf("Deleted %v objects.\n", len(output.Deleted))
 }
 return err
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is
 returned.
func (basics BucketBasics) DeleteBucket(bucketName string) error {
 _, err := basics.S3Client.DeleteBucket(context.TODO(), &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
  log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
 }
 return err
}
```

S3 バケットとオブジェクトの操作方法を示すインタラクティブなシナリオを実行します。

```


// RunGetStartedScenario is an interactive example that shows you how to use
 Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
 objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Upload a large object to the bucket by using an upload manager.
// 4. Download an object to a local file.
// 5. Download a large object by using a download manager.
// 6. Copy an object to a different folder in the bucket.
```

```
// 7. List objects in the bucket.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunGetStartedScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong with the demo.\n", r)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 getting started demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}

 count := 10
 log.Printf("Let's list up to %v buckets for your account:", count)
 buckets, err := bucketBasics.ListBuckets()
 if err != nil {
  panic(err)
 }
 if len(buckets) == 0 {
  log.Println("You don't have any buckets!")
 } else {
  if count > len(buckets) {
   count = len(buckets)
  }
  for _, bucket := range buckets[:count] {
   log.Printf("\t%v\n", *bucket.Name)
  }
 }
```

```
bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
 demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(bucketName)
if err != nil {
 panic(err)
}
if !bucketExists {
 err = bucketBasics.CreateBucket(bucketName, sdkConfig.Region)
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
const smallKey = "doc-example-key"
err = bucketBasics.UploadFile(bucketName, smallKey, smallFile)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

mibs := 30
log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your
bucket. ", mibs)
questioner.Ask("Press Enter when you're ready.")
largeBytes := make([]byte, 1024*1024*mibs)
rand.Seed(time.Now().Unix())
rand.Read(largeBytes)
largeKey := "doc-example-large"
log.Println("Uploading...")
err = bucketBasics.UploadLargeObject(bucketName, largeKey, largeBytes)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
log.Println(strings.Repeat("-", 88))
```

```
log.Printf("Let's download %v to a file.", smallKey)
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
demotools.NotEmpty{})
err = bucketBasics.DownloadFile(bucketName, smallKey, downloadFileName)
if err != nil {
 panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download the %v MiB object.", mibs)
questioner.Ask("Press Enter when you're ready.")
log.Println("Downloading...")
largeDownload, err := bucketBasics.DownloadLargeObject(bucketName, largeKey)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes.", len(largeDownload))
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(bucketName, smallKey, folderName)
if err != nil {
 panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(bucketName)
if err != nil {
 panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
for _, object := range objects {
 objKeys = append(objKeys, *object.Key)
 log.Printf("\t%v\n", *object.Key)
}
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
```

```
  "contents? (y/n)", "y") {
  log.Println("Deleting objects.")
  err = bucketBasics.DeleteObjects(bucketName, objKeys)
  if err != nil {
   panic(err)
  }
  log.Println("Deleting bucket.")
  err = bucketBasics.DeleteBucket(bucketName)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleting downloaded file %v.\n", downloadFileName)
  err = os.Remove(downloadFileName)
  if err != nil {
   panic(err)
  }
 } else {
  log.Println("Okay. Don't forget to delete objects from your bucket to avoid
 charges.")
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

- API の詳細については、『AWS SDK for Go API リファレンス』の以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Deletes the object from the Amazon S3 bucket.
 * 8. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <bucketName> <key> <objectPath> <savePath> <toBucket>

                Where:
```

```
                    bucketName - The Amazon S3 bucket to create.
                    key - The key to use.
                    objectPath - The path where the file is located (for example,
C:/AWS/book2.pdf).
                    savePath - The path where the file is saved after it's
downloaded (for example, C:/AWS/book2.pdf).
                    toBucket - An Amazon S3 bucket to where an object is copied
to (for example, C:/AWS/book2.pdf).\s
                    """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an Amazon S3 bucket.");
        createBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update a local file to the Amazon S3 bucket.");
        uploadLocalFile(s3, bucketName, key, objectPath);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Download the object to another local file.");
        getObjectBytes(s3, bucketName, key, savePath);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("4. Perform a multipart upload.");
        String multipartKey = "multiPartKey";
        multipartUpload(s3, toBucket, multipartKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List all objects located in the Amazon S3
bucket.");
        listAllObjects(s3, bucketName);
        anotherListExample(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Copy the object to another Amazon S3 bucket.");
        copyBucketObject(s3, bucketName, key, toBucket);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the object from the Amazon S3 bucket.");
        deleteObjectFromBucket(s3, bucketName, key);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Delete the Amazon S3 bucket.");
        deleteBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon S3 operations were successfully
performed");
        System.out.println(DASHES);
        s3.close();
    }

    // Create a bucket by using a S3Waiter object.
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
```

```
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteBucket(S3Client client, String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucket)
                .build();

        client.deleteBucket(deleteBucketRequest);
        System.out.println(bucket + " was deleted.");
    }

    /**
     * Upload an object in parts.
     */
    public static void multipartUpload(S3Client s3, String bucketName, String
key) {
        int mB = 1024 * 1024;
        // First create a multipart upload and get the upload id.
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object.
```

```
        UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(1).build();

        String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
                .eTag();
        CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
                .uploadId(uploadId)
                .partNumber(2).build();
        String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
                .eTag();
        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Call completeMultipartUpload operation to tell S3 to merge all
uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                .parts(part1, part2)
                .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .multipartUpload(completedMultipartUpload)
                .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        byte[] b = new byte[size];
        new Random().nextBytes(b);
```

```
        return ByteBuffer.wrap(b);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void uploadLocalFile(S3Client s3, String bucketName, String
key, String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        s3.putObject(objectRequest, RequestBody.fromFile(new File(objectPath)));
    }

    public static void listAllObjects(S3Client s3, String bucketName) {
        ListObjectsV2Request listObjectsReqManual =
ListObjectsV2Request.builder()
```

```
                .bucket(bucketName)
                .maxKeys(1)
                .build();

        boolean done = false;
        while (!done) {
            ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
            for (S3Object content : listObjResponse.contents()) {
                System.out.println(content.key());
            }

            if (listObjResponse.nextContinuationToken() == null) {
                done = true;
            }

            listObjectsReqManual = listObjectsReqManual.toBuilder()
                    .continuationToken(listObjResponse.nextContinuationToken())
                    .build();
        }
    }

    public static void anotherListExample(S3Client s3, String bucketName) {
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

        // Process response pages.
        listRes.stream()
                .flatMap(r -> r.contents().stream())
                .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

        // Helper method to work with paginated collection of items directly.
        listRes.contents().stream()
                .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

        for (S3Object content : listRes.contents()) {
            System.out.println(" Key: " + content.key() + " size = " +
content.size());
```

```
        }
    }

    public static void deleteObjectFromBucket(S3Client s3, String bucketName,
 String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        s3.deleteObject(deleteObjectRequest);
        System.out.println(key + " was deleted");
    }

    public static String copyBucketObject(S3Client s3, String fromBucket, String
 objectKey, String toBucket) {
        String encodedUrl = null;
        try {
            encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
 StandardCharsets.UTF_8.toString());
        } catch (UnsupportedEncodingException e) {
            System.out.println("URL could not be encoded: " + e.getMessage());
        }
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
                .copySource(encodedUrl)
                .destinationBucket(toBucket)
                .destinationKey(objectKey)
                .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            System.out.println("The " + objectKey + " was copied to " +
 toBucket);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

まず、必要なモジュールをすべてインポートします。

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "url";
import { readdirSync, readFileSync, writeFileSync } from "fs";

// Local helper utils.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
  GetObjectCommand,
  DeleteObjectsCommand,
```

```
  DeleteBucketCommand,
} from "@aws-sdk/client-s3";
```

前述のインポートでは、いくつかのヘルパーユーティリティを参照しています。これらのユーティリティは、このセクションの冒頭でリンクされている GitHub リポジトリのローカルにあります。参考までに、これらのユーティリティの以下の実装を参照してください。

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox } from "@inquirer/prompts";

export class Prompter {
  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}}
 options
   */
  select(options) {
    return select(options);
  }

  /**
   * @param {{ message: string }} options
   */
  input(options) {
    return input(options);
  }

  /**
   * @param {string} prompt
   */
  checkContinue = async (prompt = "") => {
    const prefix = prompt && prompt + " ";
    let ok = await this.confirm({
      message: `${prefix}Continue?`,
    });
    if (!ok) throw new Error("Exiting...");
  };

  /**
   * @param {{ message: string }} options
   */
```

```
  confirm(options) {
    return confirm(options);
  }

  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}}
 options
   */
  checkbox(options) {
    return checkbox(options);
  }
}

export const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};
```

S3 のオブジェクトは「バケット」に保存されます。新しいバケットを作成する関数を定義しましょう。

```
export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};
```

バケットには「オブジェクト」が含まれています。この関数は、ディレクトリの内容をバケットにオブジェクトとしてアップロードします。

```
export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);
  const files = keys.map((key) => {
    const filePath = `${folderPath}/${key}`;
    const fileContent = readFileSync(filePath);
    return {
```

```
      Key: key,
      Body: fileContent,
    };
  });

  for (let file of files) {
    await s3Client.send(
      new PutObjectCommand({
        Bucket: bucketName,
        Body: file.Body,
        Key: file.Key,
      }),
    );
    console.log(`${file.Key} uploaded successfully.`);
  }
};
```

オブジェクトをアップロードしたら、正しくアップロードされたことを確認します。そのためには ListObjects を使用できます。ここでは 'Key' プロパティを使用しますが、レスポンスには他にも便利なプロパティがあります。

```
export const listFilesInBucket = async ({ bucketName }) => {
  const command = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(command);
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
  console.log("\nHere's a list of files in the bucket:");
  console.log(contentsList + "\n");
};
```

バケットから別のバケットにオブジェクトをコピーしたい場合があります。そのためには、CopyObject コマンドを使用してください。

```
export const copyFileFromBucket = async ({ destinationBucket }) => {
  const proceed = await prompter.confirm({
    message: "Would you like to copy an object from another bucket?",
  });

  if (!proceed) {
    return;
  } else {
```

```
    const copy = async () => {
      try {
        const sourceBucket = await prompter.input({
          message: "Enter source bucket name:",
        });
        const sourceKey = await prompter.input({
          message: "Enter source key:",
        });
        const destinationKey = await prompter.input({
          message: "Enter destination key:",
        });

        const command = new CopyObjectCommand({
          Bucket: destinationBucket,
          CopySource: `${sourceBucket}/${sourceKey}`,
          Key: destinationKey,
        });
        await s3Client.send(command);
        await copyFileFromBucket({ destinationBucket });
      } catch (err) {
        console.error(`Copy error.`);
        console.error(err);
        const retryAnswer = await prompter.confirm({ message: "Try again?" });
        if (retryAnswer) {
          await copy();
        }
      }
    };
    await copy();
  }
};
```

バケットから複数のオブジェクトを取得するための SDK メソッドはありません。代わりに、ダウンロードして繰り返し処理するオブジェクトのリストを作成します。

```
export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
  const path = await prompter.input({
    message: "Enter destination path for files:",
  });
```

```
  for (let content of Contents) {
    const obj = await s3Client.send(
      new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
    );
    writeFileSync(
      `${path}/${content.Key}`,
      await obj.Body.transformToByteArray(),
    );
  }
  console.log("Files downloaded successfully.\n");
};
```

では、リソースをクリーンアップしましょう。バケットを削除するには、そのバケットを空にしておく必要があります。これら 2 つの関数はバケットを空にして削除します。

```
export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log(`${bucketName} deleted successfully.\n`);
};
```

「main」関数はすべてをまとめます。このファイルを直接実行すると、main 関数が呼び出されます。

```
const main = async () => {
  const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
    import.meta.url,
```

```
  )}../../../../resources/sample_files/.sample_media`;

  try {
    console.log(wrapText("Welcome to the Amazon S3 getting started example."));
    console.log("Let's create a bucket.");
    const bucketName = await createBucket();
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("File upload."));
    console.log(
      "I have some default files ready to go. You can edit the source code to
 provide your own.",
    );
    await uploadFilesToBucket({
      bucketName,
      folderPath: OBJECT_DIRECTORY,
    });

    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Copy files."));
    await copyFileFromBucket({ destinationBucket: bucketName });
    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Download files."));
    await downloadFilesFromBucket({ bucketName });

    console.log(wrapText("Clean up."));
    await emptyBucket({ bucketName });
    await deleteBucket({ bucketName });
  } catch (err) {
    console.error(err);
  }
};
```

- API の詳細については、『AWS SDK for JavaScript API リファレンス』の以下のトピックを参照してください。
  - CopyObject
  - CreateBucket

- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
        savePath - The path where the file is saved after it's downloaded (for
 example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to (for
 example, C:/AWS/book2.pdf).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val bucketName = args[0]
    val key = args[1]
    val objectPath = args[2]
```

```
    val savePath = args[3]
    val toBucket = args[4]

    // Create an Amazon S3 bucket.
    createBucket(bucketName)

    // Update a local file to the Amazon S3 bucket.
    putObject(bucketName, key, objectPath)

    // Download the object to another local file.
    getObjectFromMrap(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
```

```
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
```

```
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
```

```
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        echo("\n");
        echo("---------------------------------------\n");
        print("Welcome to the Amazon S3 getting started demo using PHP!\n");
        echo("---------------------------------------\n");

        $region = 'us-west-2';

        $this->s3client = new S3Client([
                'region' => $region,
        ]);
        /* Inline declaration example
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
        */

        $this->bucketName = "doc-example-bucket-" . uniqid();

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
```

```
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with object copying before continuing.");
        }

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
```

```
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (count($check) <= 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

```
        echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- API の詳細については、『AWS SDK for PHP API リファレンス』の以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError


def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)
```

```
    bucket_name = f"doc-example-bucket-{uuid.uuid4()}"
    bucket = s3_resource.Bucket(bucket_name)
    try:
        bucket.create(
            CreateBucketConfiguration={
                "LocationConstraint": s3_resource.meta.client.meta.region_name
            }
        )
        print(f"Created demo bucket named {bucket.name}.")
    except ClientError as err:
        print(f"Tried and failed to create demo bucket {bucket_name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")
        print(f"\nCan't continue the demo without a bucket!")
        return

    file_name = None
    while file_name is None:
        file_name = input("\nEnter a file you want to upload to your bucket: ")
        if not os.path.exists(file_name):
            print(f"Couldn't find file {file_name}. Are you sure it exists?")
            file_name = None

    obj = bucket.Object(os.path.basename(file_name))
    try:
        obj.upload_file(file_name)
        print(
            f"Uploaded file {file_name} into bucket {bucket.name} with key
 {obj.key}."
        )
    except S3UploadFailedError as err:
        print(f"Couldn't upload file {file_name} to {bucket.name}.")
        print(f"\t{err}")

    answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
    if answer.lower() == "y":
        data = io.BytesIO()
        try:
            obj.download_fileobj(data)
            data.seek(0)
            print(f"Got your object. Here are the first 20 bytes:\n")
            print(f"\t{data.read(20)}")
        except ClientError as err:
```

```
            print(f"Couldn't download {obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")

    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)?
 "
    )
    if answer.lower() == "y":
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
```

```
            )

    print("Thanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require "aws-sdk-s3"

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
```

```
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "doc-example-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: "us-east-1" # Note: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts("Tried and failed to create demo bucket.")
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end

  # Requests a file name from the user.
  #
  # @return The name of the file.
  def create_file
    File.open("demo.txt", w) { |f| f.write("This is a demo file.") }
  end

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
  # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded
file.
  def upload_file(bucket)
    File.open("demo.txt", "w+") { |f| f.write("This is a demo file.") }
    s3_object = bucket.object(File.basename("demo.txt"))
    s3_object.upload_file("demo.txt")
    puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
  rescue Aws::Errors::ServiceError => e
```

```
    puts("Couldn't upload file demo.txt to #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    s3_object
  end

  # Downloads an Amazon S3 object to a file.
  #
  # @param s3_object [Aws::S3::Object] The object to download.
  def download_file(s3_object)
    puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
    answer = gets.chomp.downcase
    if answer == "y"
      puts("Enter a name for the downloaded file: ")
      file_name = gets.chomp
      s3_object.download_file(file_name)
      puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't download #{s3_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Copies an Amazon S3 object to a subfolder within the same bucket.
  #
  # @param source_object [Aws::S3::Object] The source object to copy.
  # @return [Aws::S3::Object, nil] The destination object.
  def copy_object(source_object)
    dest_object = nil
    puts("\nDo you want to copy #{source_object.key} to a subfolder in your
 bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == "y"
      dest_object = source_object.bucket.object("demo-folder/
#{source_object.key}")
      dest_object.copy_from(source_object)
      puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
```

```
  else
    dest_object
  end

  # Lists the objects in an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to query.
  def list_objects(bucket)
    puts("\nYour bucket contains the following objects:")
    bucket.objects.each do |obj|
      puts("\t#{obj.key}")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list the objects in bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
 ")
    answer = gets.chomp.downcase
    if answer == "y"
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the Amazon S3 getting started demo!")
  puts("-" * 88)

  bucket = scenario.create_bucket
```

```
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

  puts("Thanks for watching!")
  puts("-" * 88)
rescue Aws::Errors::ServiceError
  puts("Something went wrong with the demo!")
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME
 == __FILE__
```

- API の詳細については、『AWS SDK for Ruby API リファレンス』の以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

シナリオを実行するバイナリクレートのコード。

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{config::Region, Client};
use s3_service::error::Error;
use uuid::Uuid;


#[tokio::main]
async fn main() -> Result<(), Error> {
    let (region, client, bucket_name, file_name, key, target_key) =
 initialize_variables().await;

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name,
 key, target_key).await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (Region, Client, String, String, String,
 String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);

    let bucket_name = format!("doc-example-bucket-{}", Uuid::new_v4());

    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    (region, client, bucket_name, file_name, key, target_key)
}

async fn run_s3_operations(
    region: Region,
    client: Client,
    bucket_name: String,
```

```
    file_name: String,
    key: String,
    target_key: String,
) -> Result<(), Error> {
    s3_service::create_bucket(&client, &bucket_name, region.as_ref()).await?;
    s3_service::upload_object(&client, &bucket_name, &file_name, &key).await?;
    let _object = s3_service::download_object(&client, &bucket_name, &key).await;
    s3_service::copy_object(&client, &bucket_name, &key, &target_key).await?;
    s3_service::list_objects(&client, &bucket_name).await?;
    s3_service::delete_objects(&client, &bucket_name).await?;
    s3_service::delete_bucket(&client, &bucket_name).await?;

    Ok(())
}
```

バイナリによって呼び出される共通のアクションを持つライブラリクレート。

```
use aws_sdk_s3::operation::{
    copy_object::{CopyObjectError, CopyObjectOutput},
    create_bucket::{CreateBucketError, CreateBucketOutput},
    get_object::{GetObjectError, GetObjectOutput},
    list_objects_v2::ListObjectsV2Output,
    put_object::{PutObjectError, PutObjectOutput},
};
use aws_sdk_s3::types::{
    BucketLocationConstraint, CreateBucketConfiguration, Delete,
 ObjectIdentifier,
};
use aws_sdk_s3::{error::SdkError, primitives::ByteStream, Client};
use error::Error;
use std::path::Path;
use std::str;

pub mod error;

pub async fn delete_bucket(client: &Client, bucket_name: &str) -> Result<(),
 Error> {
    client.delete_bucket().bucket(bucket_name).send().await?;
    println!("Bucket deleted");
    Ok(())
```

```
}

pub async fn delete_objects(client: &Client, bucket_name: &str) ->
 Result<Vec<String>, Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    let mut delete_objects: Vec<ObjectIdentifier> = vec![];
    for obj in objects.contents() {
        let obj_id = ObjectIdentifier::builder()
            .set_key(Some(obj.key().unwrap().to_string()))
            .build()
            .map_err(Error::from)?;
        delete_objects.push(obj_id);
    }

    let return_keys = delete_objects.iter().map(|o| o.key.clone()).collect();

    if !delete_objects.is_empty() {
        client
            .delete_objects()
            .bucket(bucket_name)
            .delete(
                Delete::builder()
                    .set_objects(Some(delete_objects))
                    .build()
                    .map_err(Error::from)?,
            )
            .send()
            .await?;
    }

    let objects: ListObjectsV2Output =
 client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(Error::unhandled(
            "There were still objects left in the bucket.",
        )),
    }
}
```

```
pub async fn list_objects(client: &Client, bucket: &str) -> Result<(), Error> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}

pub async fn copy_object(
    client: &Client,
    bucket_name: &str,
    object_key: &str,
    target_key: &str,
) -> Result<CopyObjectOutput, SdkError<CopyObjectError>> {
    let mut source_bucket_and_object: String = "".to_owned();
    source_bucket_and_object.push_str(bucket_name);
    source_bucket_and_object.push('/');
    source_bucket_and_object.push_str(object_key);

    client
        .copy_object()
        .copy_source(source_bucket_and_object)
        .bucket(bucket_name)
        .key(target_key)
        .send()
        .await
}
```

```
pub async fn download_object(
    client: &Client,
    bucket_name: &str,
    key: &str,
) -> Result<GetObjectOutput, SdkError<GetObjectError>> {
    client
        .get_object()
        .bucket(bucket_name)
        .key(key)
        .send()
        .await
}

pub async fn upload_object(
    client: &Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<PutObjectOutput, SdkError<PutObjectError>> {
    let body = ByteStream::from_path(Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
}

pub async fn create_bucket(
    client: &Client,
    bucket_name: &str,
    region: &str,
) -> Result<CreateBucketOutput, SdkError<CreateBucketError>> {
    let constraint = BucketLocationConstraint::from(region);
    let cfg = CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await
```

```
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create an Amazon Simple Storage Service (Amazon S3) bucket. "
TRY.
    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
    ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
  CATCH /aws1/cx_s3_bucketalrdyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
  CATCH /aws1/cx_s3_bktalrdyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.
```

```
"Upload an object to an S3 bucket."
TRY.
    "Get contents of file from application server."
    DATA lv_file_content TYPE xstring.
    OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
    READ DATASET iv_key INTO lv_file_content.
    CLOSE DATASET iv_key.

    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_file_content
    ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Get an object from a bucket. "
TRY.
    DATA(lo_result) = lo_s3->getobject(
                iv_bucket = iv_bucket_name
                iv_key = iv_key
            ).
    DATA(lv_object_data) = lo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
  CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" Copy an object to a subfolder in a bucket. "
TRY.
    lo_s3->copyobject(
      iv_bucket = iv_bucket_name
      iv_key = |{ iv_copy_to_folder }/{ iv_key }|
      iv_copysource = |{ iv_bucket_name }/{ iv_key }|
    ).
    MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
  CATCH /aws1/cx_s3_nosuchkey.
```

```
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " List objects in the bucket. "
    TRY.
        DATA(lo_list) = lo_s3->listobjects(
           iv_bucket = iv_bucket_name
         ).
        MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
    DATA text TYPE string VALUE 'Object List - '.
    DATA lv_object_key TYPE /aws1/s3_objectkey.
    LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
      lv_object_key = lo_object->get_key( ).
      CONCATENATE lv_object_key ', ' INTO text.
    ENDLOOP.
    MESSAGE text TYPE'I'.

    " Delete the objects in a bucket. "
    TRY.
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_key
        ).
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = |{ iv_copy_to_folder }/{ iv_key }|
        ).
        MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.


    " Delete the bucket. "
    TRY.
        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name
        ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
```

```
    ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SDK for Swift への呼び出しを処理する Swift クラス。

```
import Foundation
import AWSS3
import ClientRuntime
import AWSClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
```

```
public class ServiceHandler {
    let client: S3Client

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
 drive the AWS calls
    /// used for the example.
    ///
    /// - Returns: A new ``ServiceHandler`` object, ready to be called to
    ///            execute AWS operations.
    public init() async {
        do {
            client = try S3Client(region: "us-east-2")
        } catch {
            print("ERROR: ", dump(error, name: "Initializing S3 client"))
            exit(1)
        }
    }

    /// Create a new user given the specified name.
    ///
    /// - Parameters:
    ///   - name: Name of the bucket to create.
    /// Throws an exception if an error occurs.
    public func createBucket(name: String) async throws {
        let config = S3ClientTypes.CreateBucketConfiguration(
            locationConstraint: .usEast2
        )
        let input = CreateBucketInput(
            bucket: name,
            createBucketConfiguration: config
        )
        _ = try await client.createBucket(input: input)
    }

    /// Delete a bucket.
    /// - Parameter name: Name of the bucket to delete.
    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        _ = try await client.deleteBucket(input: input)
    }

    /// Upload a file from local storage to the bucket.
```

```
    /// - Parameters:
    ///   - bucket: Name of the bucket to upload the file to.
    ///   - key: Name of the file to create.
    ///   - file: Path name of the file to upload.
    public func uploadFile(bucket: String, key: String, file: String) async
throws {
        let fileUrl = URL(fileURLWithPath: file)
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.from(data: fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }

    /// Create a file in the specified bucket with the given name. The new
    /// file's contents are uploaded from a `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket to create a file in.
    ///   - key: Name of the file to create.
    ///   - data: A `Data` object to write into the new file.
    public func createFile(bucket: String, key: String, withData data: Data)
async throws {
        let dataStream = ByteStream.from(data: data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }

    /// Download the named file to the given directory on the local device.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket that contains the file to be copied.
    ///   - key: The name of the file to copy from the bucket.
    ///   - to: The path of the directory on the local device where you want to
    ///     download the file.
```

```
    public func downloadFile(bucket: String, key: String, to: String) async
throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the data stream object. Return immediately if there isn't one.
        guard let body = output.body,
              let data = try await body.readData() else {
            return
        }
        try data.write(to: fileUrl)
    }

    /// Read the specified file from the given S3 bucket into a Swift
    /// `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to read.
    ///   - key: Name of the file within the bucket to read.
    ///
    /// - Returns: A `Data` object containing the complete file data.
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the stream and return its contents in a `Data` object. If
        // there is no stream, return an empty `Data` object instead.
        guard let body = output.body,
              let data = try await body.readData() else {
            return "".data(using: .utf8)!
        }

        return data
    }

    /// Copy a file from one bucket to another.
```

```
    ///
    /// - Parameters:
    ///   - sourceBucket: Name of the bucket containing the source file.
    ///   - name: Name of the source file.
    ///   - destBucket: Name of the bucket to copy the file into.
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        _ = try await client.copyObject(input: input)
    }

    /// Deletes the specified file from Amazon S3.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to delete.
    ///   - key: Name of the file to delete.
    ///
    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.deleteObject(input: input)
        } catch {
            throw error
        }
    }

    /// Returns an array of strings, each naming one file in the
    /// specified bucket.
    ///
    /// - Parameter bucket: Name of the bucket to get a file listing for.
    /// - Returns: An array of `String` objects, each giving the name of
    ///            one file contained in the bucket.
    public func listBucketFiles(bucket: String) async throws -> [String] {
```

```
        let input = ListObjectsV2Input(
            bucket: bucket
        )
        let output = try await client.listObjectsV2(input: input)
        var names: [String] = []

        guard let objList = output.contents else {
            return []
        }

        for obj in objList {
            if let objName = obj.key {
                names.append(objName)
            }
        }

        return names
    }
}
```

SDK 呼び出しを管理する Swift コマンドラインプログラム。

```
import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String

    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String
```

```
    @Argument(help: "Directory where you want to download the file from the S3
bucket")
    var downloadDir: String

    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: """
        Performs the following Amazon S3 commands:

        * `CreateBucket`
        * `PutObject`
        * `GetObject`
        * `CopyObject`
        * `ListObjects`
        * `DeleteObjects`
        * `DeleteBucket`
        """
    )

    /// Called by ``main()`` to do the actual running of the AWS
    /// example.
    func runAsync() async throws {
        let serviceHandler = await ServiceHandler()

        // 1. Create the bucket.
        print("Creating the bucket \(bucketName)...")
        try await serviceHandler.createBucket(name: bucketName)

        // 2. Upload a file to the bucket.
        print("Uploading the file \(uploadSource)...")
        try await serviceHandler.uploadFile(bucket: bucketName, key: objName,
file: uploadSource)

        // 3. Download the file.
        print("Downloading the file \(objName) to \(downloadDir)...")
        try await serviceHandler.downloadFile(bucket: bucketName, key: objName,
to: downloadDir)

        // 4. Copy the file to another bucket.
        print("Copying the file to the bucket \(destBucket)...")
        try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)
```

```
        // 5. List the contents of the bucket.

        print("Getting a list of the files in the bucket \(bucketName)")
        let fileList = try await serviceHandler.listBucketFiles(bucket:
 bucketName)
        let numFiles = fileList.count
        if numFiles != 0 {
            print("\(numFiles) file\((numFiles > 1) ? "s" : "") in bucket
 \(bucketName):")
            for name in fileList {
                print("  \(name)")
            }
        } else {
            print("No files found in bucket \(bucketName)")
        }

        // 6. Delete the objects from the bucket.

        print("Deleting the file \(objName) from the bucket \(bucketName)...")
        try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
        print("Deleting the file \(objName) from the bucket \(destBucket)...")
        try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

        // 7. Delete the bucket.
        print("Deleting the bucket \(bucketName)...")
        try await serviceHandler.deleteBucket(name: bucketName)

        print("Done.")
    }
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
```

```
        }
    }
}
```

- API の詳細については、AWS SDK for Swift API リファレンスの以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 オブジェクトの暗号化の使用を開始する

次のコード例は、Amazon S3 オブジェクトの暗号化を開始する方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
```

```
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to apply client encryption to an object in an
    /// Amazon Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class SSEClientEncryption
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "exampleobject.txt";
            string copyTargetKeyName = "examplecopy.txt";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // Upload the object.
                PutObjectRequest putObjectRequest = await
UploadObjectAsync(client, bucketName, keyName, base64Key);

                // Download the object and verify that its contents match what
you uploaded.
                await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);

                // Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

                // Copy both the source and target objects using server-side
encryption with
```

```
                // an encryption key.
                await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PutObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
        /// object will be uploaded.</param>
        /// <param name="keyName">The name of the object to upload to the Amazon
S3
        /// bucket.</param>
        /// <param name="base64Key">The encryption key.</param>
        /// <returns>The PutObjectRequest object for use by
DownloadObjectAsync.</returns>
        public static async Task<PutObjectRequest> UploadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            PutObjectRequest putObjectRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };
            PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }
```

```
        /// <summary>
        /// Downloads an encrypted object from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// is located.</param>
        /// <param name="keyName">The name of the Amazon S3 object to download.</
param>
        /// <param name="base64Key">The encryption key used to encrypt the
        /// object.</param>
        /// <param name="putObjectRequest">The PutObjectRequest used to upload
        /// the object.</param>
        public static async Task DownloadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key,
            PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // Provide encryption information for the object stored in Amazon
 S3.
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            using (GetObjectResponse getResponse = await
 client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
 StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (string.Compare(putObjectRequest.ContentBody, content) == 0)
                {
                    Console.WriteLine("Object content is same as we uploaded");
```

```
                }
                else
                {
                    Console.WriteLine("Error...Object content is not same.");
                }

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                {
                    Console.WriteLine("Object encryption method is AES256, same
as we set");
                }
                else
                {
                    Console.WriteLine("Error...Object encryption method is not
the same as AES256 we set");
                }
            }
        }

        /// <summary>
        /// Retrieves the metadata associated with an Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to call GetObjectMetadataAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket containing
the
        /// object for which we want to retrieve metadata.</param>
        /// <param name="keyName">The name of the object for which we wish to
        /// retrieve the metadata.</param>
        /// <param name="base64Key">The encryption key associated with the
        /// object.</param>
        public static async Task GetObjectMetadataAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,
```

```
                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used
is: {0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }

        /// <summary>
        /// Copies an encrypted object from one Amazon S3 bucket to another.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// CopyObjectAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket containing the object
        /// to copy.</param>
        /// <param name="keyName">The name of the object to copy.</param>
        /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the
object
        /// will be copied.</param>
        /// <param name="aesEncryption">The encryption type to use.</param>
        /// <param name="base64Key">The encryption key to use.</param>
        public static async Task CopyObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string copyTargetKeyName,
            Aes aesEncryption,
            string base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
                DestinationKey = copyTargetKeyName,
```

```
                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
            };
            await client.CopyObjectAsync(copyRequest);
        }
    }
```

- API の詳細については、『AWS SDK for .NET API リファレンス』の以下のトピックを参照してください。
  - CopyObject
  - GetObject
  - GetObjectMetadata

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 オブジェクトのタグの使用を開始する

次のコード例は、Amazon S3 オブジェクトのタグの使用を開始する方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to work with tags in Amazon Simple Storage
    /// Service (Amazon S3) objects.
    /// </summary>
    public class ObjectTag
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "newobject.txt";
            string filePath = @"*** file path ***";

            // Specify your bucket region (an example region is shown).
            RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

            var client = new AmazonS3Client(bucketRegion);
            await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
        }

        /// <summary>
        /// This method uploads an object with tags. It then shows the tag
        /// values, changes the tags, and shows the new tags.
        /// </summary>
        /// <param name="client">The Initialized Amazon S3 client object used
        /// to call the methods to create and change an objects tags.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object will be stored.</param>
        /// <param name="keyName">A string representing the key name of the
        /// object to be tagged.</param>
        /// <param name="filePath">The directory location and file name of the
        /// object to be uploaded to the Amazon S3 bucket.</param>
        public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
        {
            try
            {
```

```
                // Create an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    FilePath = filePath,
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Keyx1", Value = "Value1" },
                        new Tag { Key = "Keyx2", Value = "Value2" },
                    },
                };

                PutObjectResponse response = await
client.PutObjectAsync(putRequest);

                // Now retrieve the new object's tags.
                GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

                // Display the tag values.
                objectTags.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

                Tagging newTagSet = new Tagging()
                {
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Key3", Value = "Value3" },
                        new Tag { Key = "Key4", Value = "Value4" },
                    },
                };

                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
                {
```

```
                    BucketName = bucketName,
                    Key = keyName,
                    Tagging = newTagSet,
                };

                PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

                // Retrieve the tags again and show the values.
                GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

                objectTags2.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine(
                        $"Error: '{ex.Message}'");
            }
        }
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「GetObjectTagging」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 オブジェクトのリーガルホールド設定を取得する

次のコード例は、S3 バケットのリーガルホールド設定を取得する方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
{bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
```

```
        }
    }
```

- API の詳細については、AWS SDK for .NET API リファレンスの「GetObjectLegalHold」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   // Get the legal hold details for an S3 object.
   public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
       try {
           GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

           GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
           System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
               ":\n\tStatus: " + response.legalHold().status());
           return response.legalHold();

       } catch (S3Exception ex) {
           System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
       }

       return null;
   }
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「GetObjectLegalHold」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import { fileURLToPath } from "url";
import { GetObjectLegalHoldCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new GetObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const response = await client.send(command);
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
  } catch (err) {
    console.error(err);
```

```
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "DOC-EXAMPLE-BUCKET", "OBJECT_KEY");
}
```

- API の詳細については、AWS SDK for JavaScript API リファレンスの「GetObjectLegalHold」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 オブジェクトロック機能を操作する

次のコード例は、S3 オブジェクトロック機能を使用する方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

Amazon S3 オブジェクトロック機能を示すインタラクティブなシナリオを実行します。

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;
```

```
namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
        1. Create test Amazon Simple Storage Service (S3) buckets with different
 lock policies.
        2. Upload sample objects to each bucket.
        3. Set some Legal Hold and Retention Periods on objects and buckets.
        4. Investigate lock policies by viewing settings or attempting to delete
 or overwrite objects.
        5. Clean up objects and buckets.
   */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
    public static IConfiguration _configuration = null!;
    private static string _resourcePrefix = null!;
    private static string noLockBucketName = null!;
    private static string lockEnabledBucketName = null!;
    private static string retentionAfterCreationBucketName = null!;
    private static List<string> bucketNames = new List<string>();
    private static List<string> fileNames = new List<string>();

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3ActionsWrapper>()
            )
            .Build();
```

```
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        ConfigurationSetup();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Workflow Scenario.");
            Console.WriteLine(new string('-', 80));
            await Setup(true);

            await DemoActionChoices();

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Object Locking Workflow is complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await Cleanup(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
```

```
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

        noLockBucketName = _resourcePrefix + "-no-lock";
        lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
        retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-
creation";

        bucketNames.Add(noLockBucketName);
        bucketNames.Add(lockEnabledBucketName);
        bucketNames.Add(retentionAfterCreationBucketName);
    }

    // <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Setup(bool interactive)
    {
        Console.WriteLine(
            "\nFor this workflow, we will use the AWS SDK for .NET to create
 several S3\n" +
            "buckets and files to demonstrate working with S3 locking features.
\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nS3 buckets can be created either with or without
 object lock enabled.");
        await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName,
 false);
```

```
        await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
true);
        await
_s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
false);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nA bucket can be configured to use object locking
with a default retention period.");
        await
_s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,
true,
            ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nObject lock policies can also be added to existing
buckets.");
        await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        // Upload some files to the buckets.
        Console.WriteLine("\nNow let's add some test files:");
        var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";
        int fileCount = 2;
        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a bucket.");
        }

        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileCount; i++)
```

```
            {
                var numberedFileName = Path.GetFileNameWithoutExtension(fileName)
+ i + Path.GetExtension(fileName);
                fileNames.Add(numberedFileName);
                await _s3ActionsWrapper.UploadFileAsync(bucketName,
numberedFileName, fileName);
            }
        }
        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        if (!interactive)
            return true;
        Console.WriteLine("\nNow we can set some object lock policies on
individual files:");
        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileNames.Count; i++)
            {
                // No modifications to the objects in the first bucket.
                if (bucketName != bucketNames[0])
                {
                    var exampleFileName = fileNames[i];
                    switch (i)
                    {
                        case 0:
                            {
                                var question =
                                    $"\nWould you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a legal hold.
                                    await
_s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
ObjectLockLegalHoldStatus.On);

                                }
                                break;
                            }
                        case 1:
                            {
                                var question =
```

```
                                    $"\nWould you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
                                    "\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a Governance mode retention period for
1 day.
                                    await
_s3ActionsWrapper.ModifyObjectRetentionPeriod(
                                        bucketName, exampleFileName,
                                        ObjectLockRetentionMode.Governance,
                                        DateTime.UtcNow.AddDays(1));
                                }
                                break;
                            }
                        }
                    }
                }
                Console.WriteLine(new string('-', 80));
                return true;
            }

            // <summary>
            /// List all of the current buckets and objects.
            /// </summary>
            /// <param name="interactive">True to run as interactive.</param>
            /// <returns>The list of buckets and objects.</returns>
            public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
            {
                var allObjects = new List<S3ObjectVersion>();
                foreach (var bucketName in bucketNames)
                {
                    var objectsInBucket = await
_s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
                    foreach (var objectKey in objectsInBucket.Versions)
                    {
                        allObjects.Add(objectKey);
                    }
                }
```

```
        if (interactive)
        {
            Console.WriteLine("\nCurrent buckets and objects:\n");
            int i = 0;
            foreach (var bucketObject in allObjects)
            {
                i++;
                Console.WriteLine(
                    $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
            }
        }

        return allObjects;
    }

    /// <summary>
    /// Present the user with the demo action choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<bool> DemoActionChoices()
    {
        var choices = new string[]{
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."};

        var choice = 0;
        // Keep asking the user until they choose to move on.
        while (choice != 6)
        {
            Console.WriteLine(new string('-', 80));
            choice = GetChoiceResponse(
                "\nExplore the S3 locking features by selecting one of the
following choices:"
                , choices);
            Console.WriteLine(new string('-', 80));
            switch (choice)
            {
                case 0:
```

```
                    {
                        await ListBucketsAndObjects(true);
                        break;
                    }
                case 1:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 2:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 3:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
overwrite:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        // Create the file if it does not already exist.
                        if (!File.Exists(allFiles[fileChoice].Key))
                        {
                            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
                            await sw.WriteLineAsync(
                                "This is a sample file for uploading to a
bucket.");
                        }
```

```
                        await
_s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
                        break;
                    }
                case 4:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object and
bucket to view:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        await
_s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
                        break;
                    }
                case 5:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
view:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        break;
                    }
            }
        }
        return true;
    }

    // <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
```

```
        if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
        {
            // Remove all locks and delete all buckets and objects.
            var allFiles = await ListBucketsAndObjects(false);
            foreach (var fileInfo in allFiles)
            {
                // Check for a legal hold.
                var legalHold = await
_s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
                if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
                {
                    await
_s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
ObjectLockLegalHoldStatus.Off);
                }

                // Check for a retention period.
                var retention = await
_s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);
                var hasRetentionPeriod = retention?.Mode ==
ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
DateTime.UtcNow.Date;
                await
_s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key,
hasRetentionPeriod, fileInfo.VersionId);
            }

            foreach (var bucketName in bucketNames)
            {
                await _s3ActionsWrapper.DeleteBucketByName(bucketName);
            }

        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }
```

```
        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {choices[i]}");
            }
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
```

```
        return choiceNumber - 1;
    }
}
```

S3 関数のラッパークラス。

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
 bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
 enableObjectLock)
    {
```

```
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
```

```
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
```

```
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
```

```
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in
days or years but not both.
                        }
                    }
                }
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };
```

```
            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in
{bucketName}: " +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock retention:
'{ex.Message}'");
            return new ObjectLockRetention();
        }
    }

    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
```

```
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
{bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }

    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
    public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
    {
```

```
        try
        {
            var request = new GetObjectLockConfigurationRequest()
            {
                BucketName = bucketName
            };

            var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                              $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                              $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

            return response.ObjectLockConfiguration;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }

    /// <summary>
    /// Upload a file from the local computer to an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object to
upload.</param>
    /// <returns>True if success.<returns>
    public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
            ChecksumAlgorithm = ChecksumAlgorithm.SHA256
        };
```

```
        var response = await _amazonS3.PutObjectAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"\tSuccessfully uploaded {objectName} to
 {bucketName}.");
            return true;
        }
        else
        {
            Console.WriteLine($"\tCould not upload {objectName} to
 {bucketName}.");
            return false;
        }
    }

    /// <summary>
    /// List bucket objects and versions.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <returns>The list of objects and versions.</returns>
    public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
 bucketName)
    {
        var request = new ListVersionsRequest()
        {
            BucketName = bucketName
        };

        var response = await _amazonS3.ListVersionsAsync(request);
        return response;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
 objectKey, bool hasRetention, string? versionId = null)
```

```
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteBucketByName(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete bucket {bucketName}: " +
 ex.Message);
            return false;
        }

    }

}
```

- API の詳細については、『AWS SDK for .NET API リファレンス』の以下のトピックを参照してください。
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

Amazon S3 オブジェクトロック機能を示すインタラクティブなシナリオを実行します。

```
// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
 questioner demotools.IQuestioner
 resources  Resources
 s3Actions  *actions.S3Actions
```

```
 sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner) ObjectLockScenario {
 scenario := ObjectLockScenario{
  questioner: questioner,
  resources:  Resources{},
  s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
  sdkConfig:  sdkConfig,
 }
 scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
 scenario.resources.init(scenario.s3Actions, questioner)
 return scenario
}

type nameLocked struct {
 name   string
 locked bool
}

var createInfo = []nameLocked{
 {"standard-bucket", false},
 {"lock-bucket", true},
 {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
 log.Println("Let's create some S3 buckets to use for this workflow.")
 success := false
 for !success {
  prefix := scenario.questioner.Ask(
   "This example creates three buckets. Enter a prefix to name your buckets
 (remember bucket names must be globally unique):")

  for _, info := range createInfo {
   bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx,
 fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
   if err != nil {
    switch err.(type) {
    case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
     log.Printf("Couldn't create bucket %s.\n", bucketName)
```

```
    default:
     panic(err)
    }
    break
   }
   scenario.resources.demoBuckets[info.name] = &DemoBucket{
    name:       bucketName,
    objectKeys: []string{},
   }
   log.Printf("Created bucket %s.\n", bucketName)
  }

  if len(scenario.resources.demoBuckets) < len(createInfo) {
   scenario.resources.deleteBuckets(ctx)
  } else {
   success = true
  }
 }

 log.Println("S3 buckets created.")
 log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
 log.Println("\nA bucket can be configured to use object locking.")
 scenario.questioner.Ask("Press Enter to continue.")

 var err error
 bucket := scenario.resources.demoBuckets["retention-bucket"]
 err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
  default:
   panic(err)
  }
 } else {
  log.Printf("Object locking enabled on bucket %s.", bucket.name)
 }

 log.Println(strings.Repeat("-", 88))
}
```

```
// SetDefaultRetentionPolicy sets a default retention governance policy on a
 bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx
 context.Context) {
 log.Println("\nA bucket can be configured to use object locking with a default
 retention period.")

 bucket := scenario.resources.demoBuckets["retention-bucket"]
 retentionPeriod := scenario.questioner.AskInt("Enter the default retention
 period in days: ")
 err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
 bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
 types.ObjectLockRetentionModeGovernance)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't configure a default retention period on bucket %s.\n",
 bucket.name)
  default:
   panic(err)
  }
 } else {
  log.Printf("Default retention policy set on bucket %s with %d day retention
 period.", bucket.name, retentionPeriod)
  bucket.retentionEnabled = true
 }

 log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
 log.Println("Uploading test objects to S3 buckets.")

 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  for i := 0; i < 2; i++ {
   key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
 fmt.Sprintf("example-%d", i),
    fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
   if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
```

```
     log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
    default:
     panic(err)
    }
   } else {
    log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
    bucket.objectKeys = append(bucket.objectKeys, key)
   }
  }
 }

 scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
 log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test
 objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx
 context.Context) {
 log.Println("Now let's set object lock configurations on individual objects.")

 buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
 scenario.resources.demoBuckets["retention-bucket"]}
 for _, bucket := range buckets {
  for index, objKey := range bucket.objectKeys {
   switch index {
   case 0:
    if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold
 to %s in %s (y/n)? ", objKey, bucket.name), "y") {
     err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
 types.ObjectLockLegalHoldStatusOn)
     if err != nil {
      switch err.(type) {
      case *types.NoSuchKey:
       log.Printf("Couldn't set legal hold on %s.\n", objKey)
      default:
       panic(err)
      }
     } else {
      log.Printf("Legal hold set on %s.\n", objKey)
     }
    }
   case 1:
```

```
    q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to
 %s in %s?\n"+
     "Reminder: Only a user with the s3:BypassGovernanceRetention permission is
 able to delete this object\n"+
     "or its bucket until the retention period has expired. (y/n) ", objKey,
 bucket.name)
    if scenario.questioner.AskBool(q, "y") {
     err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
 types.ObjectLockRetentionModeGovernance, 1)
     if err != nil {
      switch err.(type) {
      case *types.NoSuchKey:
       log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
 bucket.name)
      default:
       panic(err)
      }
     } else {
      log.Printf("Retention period set to 1 for %s.", objKey)
      bucket.retentionEnabled = true
     }
    }
   }
  }
 }
 log.Println(strings.Repeat("-", 88))
}

const (
 ListAll = iota
 DeleteObject
 DeleteRetentionObject
 OverwriteObject
 ViewRetention
 ViewLegalHold
 Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
 object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
 log.Println("Now you can interact with the objects to explore the object lock
 configurations.")
 interactiveChoices := []string{
```

```
 "List all objects and buckets.",
 "Attempt to delete an object.",
 "Attempt to delete an object with retention period bypass.",
 "Attempt to overwrite a file.",
 "View the retention settings for an object.",
 "View the legal hold settings for an object.",
 "Finish the workflow."}

choice := ListAll
for choice != Finish {
 objList := scenario.GetAllObjects(ctx)
 objChoices := scenario.makeObjectChoiceList(objList)
 choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
interactiveChoices)
 switch choice {
 case ListAll:
  log.Println("The current objects in the example buckets are:")
  for _, objChoice := range objChoices {
   log.Println("\t", objChoice)
  }
 case DeleteObject, DeleteRetentionObject:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
delete:\n", objChoices)
  obj := objList[objChoice]
  deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
obj.versionId, choice == DeleteRetentionObject)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Println("Nothing to delete.")
   default:
    panic(err)
   }
  } else if deleted {
   log.Printf("Object %s deleted.\n", obj.key)
  }
 case OverwriteObject:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
  obj := objList[objChoice]
  _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
  if err != nil {
   switch err.(type) {
```

```
    case *types.NoSuchBucket:
     log.Println("Couldn't upload to nonexistent bucket.")
    default:
     panic(err)
    }
   } else {
    log.Printf("Uploaded new content to object %s.\n", obj.key)
   }
  case ViewRetention:
   objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
   obj := objList[objChoice]
   retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket,
obj.key)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Can't get retention configuration for %s.\n", obj.key)
    default:
     panic(err)
    }
   } else if retention != nil {
    log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
retention.Mode, retention.RetainUntilDate)
   } else {
    log.Printf("Object %s does not have object retention configured.\n", obj.key)
   }
  case ViewLegalHold:
   objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
   obj := objList[objChoice]
   legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket,
obj.key, obj.versionId)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
    default:
     panic(err)
    }
   } else if legalHold != nil {
    log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
   } else {
    log.Printf("Object %s does not have legal hold configured.", obj.key)
```

```
   }
  case Finish:
   log.Println("Let's clean up.")
  }
  log.Println(strings.Repeat("-", 88))
 }
}

type BucketKeyVersionId struct {
 bucket    string
 key       string
 versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns
 them in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
 []BucketKeyVersionId {
 var objectList []BucketKeyVersionId
 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Printf("Couldn't get object versions for %s.\n", bucket.name)
   default:
    panic(err)
   }
  } else {
   for _, version := range versions {
    objectList = append(objectList,
     BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
 *version.VersionId})
   }
  }
 }
 return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
 are displayed
// as choices.
```

```
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
 []BucketKeyVersionId) []string {
 choices := make([]string, len(bucketObjects))
 for i := 0; i < len(bucketObjects); i++ {
  choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
   bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
 }
 return choices
}

// Run runs the S3 Object Lock workflow scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := scenario.questioner.(*demotools.MockQuestioner)
   if isMock || scenario.questioner.AskBool("Do you want to see the full error
 message (y/n)?", "y") {
    log.Println(r)
   }
   scenario.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 Object Lock Workflow Scenario.")
 log.Println(strings.Repeat("-", 88))

 scenario.CreateBuckets(ctx)
 scenario.EnableLockOnBucket(ctx)
 scenario.SetDefaultRetentionPolicy(ctx)
 scenario.UploadTestObjects(ctx)
 scenario.SetObjectLockConfigurations(ctx)
 scenario.InteractWithObjects(ctx)

 scenario.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

この例で使用されている S3 アクションをラップする構造体を定義します。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// CreateBucketWithLock creates a new S3 bucket with optional object locking
 enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
```

```
 }

 return bucket, err
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}
```

```
// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }

 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
  lockConfig = output.ObjectLockConfiguration
 }

 return lockConfig, err
}



// GetObjectRetention retrieves the object retention configuration for an S3
 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
```

```
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
 {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }
```

```
 return err
}



// ModifyDefaultBucketRetention modifies the default retention period of an
 existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
 string) error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
```

```
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
```

```
    RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
   },
   BypassGovernanceRetention: aws.Bool(true),
  }

  _, err := actor.S3Client.PutObjectRetention(ctx, input)
  if err != nil {
   var noKey *types.NoSuchKey
   if errors.As(err, &noKey) {
    log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
    err = noKey
   }
  }

  return err
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
 string, contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
 bucket)
```

```
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
 string, versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
```

```
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
    }
   }
  }
 } else {
  deleted = true
 }
 return deleted, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
```

```
    Quiet:   aws.Bool(true),
   },
  }
  if bypassGovernance {
   input.BypassGovernanceRetention = aws.Bool(true)
  }
  delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
  if err != nil || len(delOut.Errors) > 0 {
   log.Printf("Error deleting objects from bucket %s.\n", bucket)
   if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
     log.Printf("Bucket %s does not exist.\n", bucket)
     err = noBucket
    }
   } else if len(delOut.Errors) > 0 {
    for _, outErr := range delOut.Errors {
     log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
    }
    err = fmt.Errorf("%s", *delOut.Errors[0].Message)
   }
  }
  return err
}
```

リソースをクリーンアップします。

```
// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
 name             string
 legalHold        bool
 retentionEnabled bool
 objectKeys       []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario
 and handles
// cleanup when the scenario finishes.
type Resources struct {
```

```
 demoBuckets map[string]*DemoBucket

 s3Actions  *actions.S3Actions
 questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
 demotools.IQuestioner) {
 resources.s3Actions = s3Actions
 resources.questioner = questioner
 resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 " +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if !wantDelete {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
  return
 }

 log.Println("Removing objects from S3 buckets and deleting buckets...")
 resources.deleteBuckets(ctx)
 //resources.deleteRetentionObjects(resources.retentionBucket,
 resources.retentionObjects)

 log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
 ObjectLockScenario.
```

```
func (resources *Resources) deleteBuckets(ctx context.Context) {
 for _, info := range createInfo {
  bucket := resources.demoBuckets[info.name]
  resources.deleteObjects(ctx, bucket)
  _, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
   Bucket: aws.String(bucket.name),
  })
  if err != nil {
   panic(err)
  }
 }
 resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket
 *DemoBucket) {
 lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx,
 bucket.name)
 if err != nil {
  panic(err)
 }
 versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("No objects to get from %s.\n", bucket.name)
  default:
   panic(err)
  }
 }
 delObjects := make([]types.ObjectIdentifier, len(versions))
 for i, version := range versions {
  if lockConfig != nil && lockConfig.ObjectLockEnabled ==
 types.ObjectLockEnabledEnabled {
   status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
 *version.Key, *version.VersionId)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Couldn't determine legal hold status for %s in %s.\n",
 *version.Key, bucket.name)
    default:
     panic(err)
```

```
    }
   } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
    err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
 *version.VersionId, types.ObjectLockLegalHoldStatusOff)
    if err != nil {
     switch err.(type) {
     case *types.NoSuchKey:
      log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
 bucket.name)
     default:
      panic(err)
     }
    }
   }
  }
  delObjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
 version.VersionId}
 }
 err = resources.s3Actions.DeleteObjects(ctx, bucket.name, delObjects,
 bucket.retentionEnabled)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Println("Nothing to delete.")
  default:
   panic(err)
  }
 }
}
```

- API の詳細については、『AWS SDK for Go API リファレンス』の以下のトピックを参照してください。
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

Amazon S3 オブジェクトロック機能を示すインタラクティブなシナリオを実行します。

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
 Before running this Java V2 code example, set up your development
 environment, including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

 This Java example performs the following tasks:
    1. Create test Amazon Simple Storage Service (S3) buckets with different lock
 policies.
    2. Upload sample objects to each bucket.
    3. Set some Legal Hold and Retention Periods on objects and buckets.
    4. Investigate lock policies by viewing settings or attempting to delete or
 overwrite objects.
    5. Clean up objects and buckets.
 */
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
```

```
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        // Get the current date and time to ensure bucket name is unique.
        LocalDateTime currentTime = LocalDateTime.now();

        // Format the date and time as a string.
        DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
        String timeStamp = currentTime.format(formatter);

        s3LockActions = new S3LockActions();
        bucketName = "bucket"+timeStamp;
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Workflow Scenario.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
```

```
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by
selecting one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
                    System.out.println("Enter the number of the object to
delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
```

```
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
                }

                case 2 -> {
                    System.out.println("Enter the number of the object to
delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
                }

                case 3 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();

                    // Attempt to overwrite the file.
                    try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
                        writer.write("This is a modified text.");

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
```

```
                    s3LockActions.uploadFile(bucketName, objectKey, objectKey);
                }

                case 4 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectRetention(bucketName, objectKey);
                }

                case 5 -> {
                    System.out.println("Enter the number of the object to
view:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectLegalHold(bucketName, objectKey);
                    s3LockActions.getBucketObjectLockConfiguration(bucketName);
                }

                case 6 -> {
                    System.out.println("Exiting the workflow...");
                    return;
                }

                default -> {
                    System.out.println("Invalid choice. Please select again.");
                }
            }
        }
    }
```

```
   // Clean up the resources from the scenario.
   private static void cleanup() {
       List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
       for (S3InfoObject fileInfo : allFiles) {
           String bucketName = fileInfo.getBucketName();
           String key = fileInfo.getKeyName();
           String version = fileInfo.getVersion();
           if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
               ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
               if (legalHold != null) {
                   String holdStatus = legalHold.status().name();
                   System.out.println(holdStatus);
                   if (holdStatus.compareTo("ON") == 0) {
                       s3LockActions.modifyObjectLegalHold(bucketName, key,
false);
                   }
               }
               // Check for a retention period.
               ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
               boolean hasRetentionPeriod ;
               hasRetentionPeriod = retention != null;
               s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

           } else {
               System.out.println(bucketName +" objects do not have a legal
lock");
               s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
           }
       }

       // Delete the buckets.
       System.out.println("Delete "+bucketName);
       for (String bucket : bucketNames){
           s3LockActions.deleteBucketByName(bucket);
       }
   }

   private static void setup() {
```

```
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                For this workflow, we will use the AWS SDK for Java to create
several S3
                buckets and files to demonstrate working with S3 locking
features.
                """);

        System.out.println("S3 buckets can be created either with or without
object lock enabled.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();

        // Create three S3 buckets.
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
        s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Bucket "+bucketNames.get(2) +" will be configured to
use object locking with a default retention period.");
        s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
        s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        // Upload some files to the buckets.
        System.out.println("Now let's add some test files:");
        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
                    fileNameWithoutExtension =
fileNameWithoutExtension.substring(0, extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }

        String question = null;
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println("Now we can set some object lock policies on
individual files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){

                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                            System.out.println(question);
                            String ans = scanner.nextLine().trim();
                            if (ans.equalsIgnoreCase("y")) {
                                System.out.println("**** You have selected to put
a legal hold " + exampleFileName);

                                // Set a legal hold.
                                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                            }
```

```
                        }
                        case 1 -> {
                            """
                                Would you like to add a 1 day Governance
 retention period to %s in %s (y/n)?
                                Reminder: Only a user with the
 s3:BypassGovernanceRetention permission will be able to delete this file or its
 bucket until the retention period has expired.
                                """.formatted(exampleFileName, bucketName);
                            System.out.println(question);
                            String ans2 = scanner.nextLine().trim();
                            if (ans2.equalsIgnoreCase("y")) {

 s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                            }
                        }
                    }
                }
            }
        }
    }

    // Get file extension.
    private static String getFileExtension(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        if (dotIndex > 0) {
            return fileName.substring(dotIndex);
        }
        return "";
    }

    public static void configurationSetup() {
        String noLockBucketName = bucketName + "-no-lock";
        String lockEnabledBucketName = bucketName + "-lock-enabled";
        String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
        bucketNames.add(noLockBucketName);
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
```

```
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }

        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid
 number.");
            }
        }

        return choiceNumber - 1;
    }
}
```

S3 関数のラッパークラス。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
 software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
```

```
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
 software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
 {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);
```

```
        // Convert the Instant to a ZonedDateTime object with a specific time
zone.
        ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }

    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
```

```
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
        }

        return null;
    }

    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
```

```
                }))
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the
updated value.
                if (interactive) {
                    System.out.println(i + ": "+ s3InfoObject.getKeyName());
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();
```

```
        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();


getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName
+".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
```

```
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on
"+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
        }
    }

    public void uploadFile(String bucketName, String objectName, String filePath)
{
        Path file = Paths.get(filePath);
        PutObjectRequest request = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectName)
            .checksumAlgorithm(ChecksumAlgorithm.SHA256)
            .build();

        PutObjectResponse response = getClient().putObject(request, file);
        if (response != null) {
            System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
        } else {
            System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
        }
    }

    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
```

```
        }

        PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey,
boolean hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
```

```
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }

    public void deleteBucketByName(String bucketName) {
        try {
            DeleteBucketRequest request = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

            getClient().deleteBucket(request);
            System.out.println(bucketName +" was deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
```

```
        System.out.println("\tEnabled:
 "+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
 response.objectLockConfiguration().rule().defaultRetention());
    }
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

index.js - ワークフローのエントリポイント。これにより、すべてのステップがオーケストレーションされます。GitHub にアクセスして、Scenario、ScenarioInput、ScenarioOutput、ScenarioAction の実装の詳細を確認します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
```

```
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "./welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
  confirmSetLegalHoldFileRetention,
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
  updateRetention,
  updateRetentionAction,
} from "./setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Object Locking - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
```

```
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
      ],
      initialState,
    ),
    demo: new scenarios.Scenario(
      "S3 Object Locking - Demo",
      [loadState, replAction(scenarios, client)],
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Object Locking - Destroy",
      [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};

// Call function if run directly
```

```
import { fileURLToPath } from "url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios);
}
```

welcome.steps.js - ウェルカムメッセージをコンソールに出力します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    `Welcome to the Amazon Simple Storage Service (S3) Object Locking Workflow
 Scenario. For this workflow, we will use the AWS SDK for JavaScript to create
 several S3 buckets and files to demonstrate working with S3 locking features.`,
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

setup.steps.js - バケット、オブジェクト、ファイル設定をデプロイします。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const bucketPrefix = "js-object-locking";

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    `The following buckets will be created:
         ${bucketPrefix}-no-lock with object lock False.
         ${bucketPrefix}-lock-enabled with object lock True.
         ${bucketPrefix}-retention-after-creation with object lock False.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
```

```
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const noLockBucketName = `${bucketPrefix}-no-lock`;
    const lockEnabledBucketName = `${bucketPrefix}-lock-enabled`;
    const retentionBucketName = `${bucketPrefix}-retention-after-creation`;

    await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
    await client.send(
      new CreateBucketCommand({
        Bucket: lockEnabledBucketName,
        ObjectLockEnabledForBucket: true,
      }),
    );
    await client.send(new CreateBucketCommand({ Bucket: retentionBucketName }));

    state.noLockBucketName = noLockBucketName;
    state.lockEnabledBucketName = lockEnabledBucketName;
    state.retentionBucketName = retentionBucketName;
  });

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    `The following test files will be created:
         file0.txt in ${bucketPrefix}-no-lock.
         file1.txt in ${bucketPrefix}-no-lock.
         file0.txt in ${bucketPrefix}-lock-enabled.
         file1.txt in ${bucketPrefix}-lock-enabled.
         file0.txt in ${bucketPrefix}-retention-after-creation.
         file1.txt in ${bucketPrefix}-retention-after-creation.`,
    { preformatted: true },
```

```
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    await client.send(
      new PutObjectCommand({
        Bucket: state.noLockBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.noLockBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.lockEnabledBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
```

```
        Bucket: state.lockEnabledBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.retentionBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.retentionBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 */
const updateRetention = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateRetention",
    `A bucket can be configured to use object locking with a default retention
 period.
   A default retention period will be configured for ${bucketPrefix}-retention-
after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateRetention",
    "Configure default retention period?",
```

```
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
    await client.send(
      new PutBucketVersioningCommand({
        Bucket: state.retentionBucketName,
        VersioningConfiguration: {
          MFADelete: MFADeleteStatus.Disabled,
          Status: BucketVersioningStatus.Enabled,
        },
      }),
    );

    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.retentionBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
          Rule: {
            DefaultRetention: {
              Mode: "GOVERNANCE",
              Years: 1,
            },
          },
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateLockPolicy",
    `Object lock policies can also be added to existing buckets.
   An object lock policy will be added to ${bucketPrefix}-lock-enabled.`,
    { preformatted: true },
```

```
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateLockPolicy = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateLockPolicy",
    "Add object lock policy?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateLockPolicyAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.lockEnabledBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileEnabled",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.lockEnabledBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
```

```
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileEnabledAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileEnabled",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.lockEnabledBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
 able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
```

```
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileEnabledAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.lockEnabledBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileRetention",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.retentionBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
```

```
  new scenarios.ScenarioAction(
    "setLegalHoldFileRetentionAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileRetention",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.retentionBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
 able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileRetentionAction",
    async (state) => {
      const retentionDate = new Date();
```

```
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
          BypassGovernanceRetention: true,
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.retentionBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },
  );

export {
  createBuckets,
  confirmCreateBuckets,
  createBucketsAction,
  populateBuckets,
  confirmPopulateBuckets,
  populateBucketsAction,
  updateRetention,
  confirmUpdateRetention,
  updateRetentionAction,
  updateLockPolicy,
  confirmUpdateLockPolicy,
  updateLockPolicyAction,
  confirmSetLegalHoldFileEnabled,
  setLegalHoldFileEnabledAction,
  confirmSetRetentionPeriodFileEnabled,
  setRetentionPeriodFileEnabledAction,
  confirmSetLegalHoldFileRetention,
  setLegalHoldFileRetentionAction,
  confirmSetRetentionPeriodFileRetention,
  setRetentionPeriodFileRetentionAction,
};
```

repl.steps.js - バケット内のファイルを表示および削除します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  ChecksumAlgorithm,
  DeleteObjectCommand,
  GetObjectLegalHoldCommand,
  GetObjectLockConfigurationCommand,
  GetObjectRetentionCommand,
  ListObjectVersionsCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  DELETE_FILE: 2,
  DELETE_FILE_WITH_RETENTION: 3,
  OVERWRITE_FILE: 4,
  VIEW_RETENTION_SETTINGS: 5,
  VIEW_LEGAL_HOLD_SETTINGS: 6,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    `Explore the S3 locking features by selecting one of the following choices`,
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
```

```
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
        },
        { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
        {
          name: "View the object and bucket retention settings for a file.",
          value: choices.VIEW_RETENTION_SETTINGS,
        },
        {
          name: "View the legal hold settings for a file.",
          value: choices.VIEW_LEGAL_HOLD_SETTINGS,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key, VersionId } = version;
      files.push({ bucket, key: Key, version: VersionId });
    }
  }

  return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
```

```
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.noLockBucketName,
        state.lockEnabledBucketName,
        state.retentionBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
              file.version
            })`,
            value: index,
          })),
        },
      );

      const { replChoice } = state;

      switch (replChoice) {
        case choices.LIST_ALL_FILES: {
          const files = await getAllFiles(client, [
            state.noLockBucketName,
            state.lockEnabledBucketName,
            state.retentionBucketName,
          ]);
          state.replOutput = files
            .map(
              (file) =>
                `${file.bucket}: ${file.key} (version: ${file.version})`,
            )
            .join("\n");
          break;
        }
        case choices.DELETE_FILE: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
```

```
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.DELETE_FILE_WITH_RETENTION: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
                BypassGovernanceRetention: true,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.OVERWRITE_FILE: {
          /** @type {number} */
          const fileToOverwrite = await fileInput.handle(state);
          const selectedFile = files[fileToOverwrite];
          try {
            await client.send(
              new PutObjectCommand({
                Bucket: selectedFile.bucket,
```

```
                Key: selectedFile.key,
                Body: "New content",
                ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
              }),
            );
            state.replOutput = `Overwrote ${selectedFile.key} in
 ${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to overwrite object ${selectedFile.key} in
 bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.VIEW_RETENTION_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const retention = await client.send(
              new GetObjectRetentionCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            const bucketConfig = await client.send(
              new GetObjectLockConfigurationCommand({
                Bucket: selectedFile.bucket,
              }),
            );
            state.replOutput = `Object retention for ${selectedFile.key}
 in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
 ${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
 ${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
          } catch (err) {
            state.replOutput = `Unable to fetch object lock retention:
 '${err.message}'`;
          }
          break;
        }
        case choices.VIEW_LEGAL_HOLD_SETTINGS: {
```

```
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const legalHold = await client.send(
              new GetObjectLegalHoldCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Object legal hold for ${selectedFile.key} in
 ${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
          } catch (err) {
            state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
          }
          break;
        }
        default:
          throw new Error(`Invalid replChoice: ${replChoice}`);
      }
    },
    {
      whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new scenarios.ScenarioOutput(
          "REPL output",
          (state) => state.replOutput,
          { preformatted: true },
        ),
      },
    },
  );

export { replInput, replAction, choices };
```

clean.steps.js - 作成されたすべてのリソースを破棄します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
```

```
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
  GetObjectLegalHoldCommand,
  GetObjectRetentionCommand,
  PutObjectLegalHoldCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
      state;

    const buckets = [
      noLockBucketName,
      lockEnabledBucketName,
      retentionBucketName,
    ];

    for (const bucket of buckets) {
      /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
      let objectsResponse;

      try {
```

```
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Object's bucket has already been deleted.");
          continue;
        } else {
          throw e;
        }
      }

      for (const version of objectsResponse.Versions || []) {
        const { Key, VersionId } = version;

        try {
          const legalHold = await client.send(
            new GetObjectLegalHoldCommand({
              Bucket: bucket,
              Key,
              VersionId,
            }),
          );

          if (legalHold.LegalHold?.Status === "ON") {
            await client.send(
              new PutObjectLegalHoldCommand({
                Bucket: bucket,
                Key,
                VersionId,
                LegalHold: {
                  Status: "OFF",
                },
              }),
            );
          }
        } catch (err) {
          console.log(
            `Unable to fetch legal hold for ${Key} in ${bucket}:
'${err.message}'`,
          );
        }
```

```
        try {
          const retention = await client.send(
            new GetObjectRetentionCommand({
              Bucket: bucket,
              Key,
              VersionId,
            }),
          );

          if (retention.Retention?.Mode === "GOVERNANCE") {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
                BypassGovernanceRetention: true,
              }),
            );
          }
        } catch (err) {
          console.log(
            `Unable to fetch object lock retention for ${Key} in ${bucket}:
 '${err.message}'`,
          );
        }

        await client.send(
          new DeleteObjectCommand({
            Bucket: bucket,
            Key,
            VersionId,
          }),
        );
      }

      await client.send(new DeleteBucketCommand({ Bucket: bucket }));
      console.log(`Delete for ${bucket} complete.`);
    }
  });

export { confirmCleanup, cleanupAction };
```

- API の詳細については、『AWS SDK for JavaScript API リファレンス』の以下のトピックを参照してください。
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3バケットのアクセスコントロールリスト (ACL) を管理する

次のコード例は、Amazon S3 バケットのアクセスコントロールリスト (ACL) を管理する方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to manage Amazon Simple Storage Service
/// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
```

```
    /// access.
    /// </summary>
    public class ManageACLs
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket1";
            string newBucketName = "doc-example-bucket2";
            string keyName = "sample-object.txt";
            string emailAddress = "someone@example.com";

            // If the AWS Region where your bucket is located is different from
            // the Region defined for the default user, pass the Amazon S3
bucket's
            // name to the client constructor. It should look like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
            IAmazonS3 client = new AmazonS3Client();

            await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
ACL
        /// information and then adds a new ACL to one of the objects in the
        /// Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// methods to create a bucket, get an ACL, and add a different ACL to
        /// one of the objects.</param>
        /// <param name="bucketName">A string representing the original Amazon S3
        /// bucket name.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new bucket that will be created.</param>
        /// <param name="keyName">A string representing the key name of an Amazon
S3
        /// object for which we will change the ACL.</param>
        /// <param name="emailAddress">A string representing the email address
        /// belonging to the person to whom access to the Amazon S3 bucket will
be
        /// granted.</param>
        public static async Task TestBucketObjectACLsAsync(
```

```
            IAmazonS3 client,
            string bucketName,
            string newBucketName,
            string keyName,
            string emailAddress)
        {
            try
            {
                // Create a new Amazon S3 bucket and specify canned ACL.
                var success = await CreateBucketWithCannedACLAsync(client,
 newBucketName);

                // Get the ACL on a bucket.
                await GetBucketACLAsync(client, bucketName);

                // Add (replace) the ACL on an object in a bucket.
                await AddACLToExistingObjectAsync(client, bucketName, keyName,
 emailAddress);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Exception: {amazonS3Exception.Message}");
            }
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL attached.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new Amazon S3 bucket.</param>
        /// <returns>Returns a boolean value indicating success or failure.</
returns>
        public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
 client, string newBucketName)
        {
            var request = new PutBucketRequest()
            {
                BucketName = newBucketName,
                BucketRegion = S3Region.EUWest1,

                // Add a canned ACL.
                CannedACL = S3CannedACL.LogDeliveryWrite,
```

```
            };

            var response = await client.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }


        /// <summary>
        /// Retrieves the ACL associated with the Amazon S3 bucket name in the
        /// bucketName parameter.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket for which we want to
 get the
        /// ACL list.</param>
        /// <returns>Returns an S3AccessControlList returned from the call to
        /// GetACLAsync.</returns>
        public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
 client, string bucketName)
        {
            GetACLResponse response = await client.GetACLAsync(new GetACLRequest
            {
                BucketName = bucketName,
            });

            return response.AccessControlList;
        }



        /// <summary>
        /// Adds a new ACL to an existing object in the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
 S3
        /// bucket containing the object to which we want to apply a new ACL.</
param>
        /// <param name="keyName">A string representing the name of the object
        /// to which we want to apply the new ACL.</param>
        /// <param name="emailAddress">The email address of the person to whom
        /// we will be applying to whom access will be granted.</param>
```

```
        public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
 string bucketName, string keyName, string emailAddress)
        {
            // Retrieve the ACL for an object.
            GetACLResponse aclResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
            });

            S3AccessControlList acl = aclResponse.AccessControlList;

            // Retrieve the owner.
            Owner owner = acl.Owner;

            // Clear existing grants.
            acl.Grants.Clear();

            // Add a grant to reset the owner's full permission
            // (the previous clear statement removed all permissions).
            var fullControlGrant = new S3Grant
            {
                Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
            };
            acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

            // Specify email to identify grantee for granting permissions.
            var grantUsingEmail = new S3Grant
            {
                Grantee = new S3Grantee { EmailAddress = emailAddress },
                Permission = S3Permission.WRITE_ACP,
            };

            // Specify log delivery group as grantee.
            var grantLogDeliveryGroup = new S3Grant
            {
                Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
                Permission = S3Permission.WRITE,
            };

            // Create a new ACL.
            var newAcl = new S3AccessControlList
```

```
            {
                Grants = new List<S3Grant> { grantUsingEmail,
grantLogDeliveryGroup },
                Owner = owner,
            };

            // Set the new ACL. We're throwing away the response here.
            _ = await client.PutACLAsync(new PutACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
                AccessControlList = newAcl,
            });
        }

    }
```

- API の詳細については、『AWS SDK for .NET API リファレンス』の以下のトピックを参照してください。
  - GetBucketAcl
  - GetObjectAcl
  - PutBucketAcl
  - PutObjectAcl

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して、バージョン管理されている Amazon S3 オブジェクトを Lambda 関数でバッチで管理する

次のコード例は、バージョン管理されている S3 オブジェクトを Lambda 関数でバッチで管理する方法を示しています。

Python

SDK for Python (Boto3)

AWS Lambda 関数を呼び出して処理を実行するジョブを作成することによって、Amazon Simple Storage Service (Amazon S3) のバージョン管理されているオブジェクトをバッチで操作する方法を示しています。この例では、バージョン対応のバケットを作成し、Lewis Carroll の詩「You Are Old, Father William」からスタンザをアップロードし、Amazon S3 のバッチジョブを使用して、さまざまな方法で詩をひねります。

以下ではその方法を説明しています。

- バージョン管理されたオブジェクトを操作する Lambda 関数を作成します。
- 更新するオブジェクトのマニフェストを作成します。
- Lambda 関数を呼び出してオブジェクトを更新するバッチジョブを作成します。
- Lambda 関数を削除します。
- バージョン管理されているバケットを空にして削除します。

この例は GitHub で最もよく確認できます。完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon S3

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 の URI を解析する

次のコード例は、Amazon S3 の URI を解析してバケット名やオブジェクトキーなどの重要なコンポーネントを抽出する方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

S3Uri クラスを使用して Amazon S3 の URI を解析します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

    /**
     *
     * @param s3Client    - An S3Client through which you acquire an S3Uri
 instance.
     * @param s3ObjectUrl - A complex URL (String) that is used to demonstrate
 S3Uri
     *                    capabilities.
     */
    public static void parseS3UriExample(S3Client s3Client, String s3ObjectUrl) {
        logger.info(s3ObjectUrl);
        // Console output:
        // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

        // Create an S3Utilities object using the configuration of the s3Client.
        S3Utilities s3Utilities = s3Client.utilities();

        // From a String URL create a URI object to pass to the parseUri()
 method.
        URI uri = URI.create(s3ObjectUrl);
```

```
        S3Uri s3Uri = s3Utilities.parseUri(uri);

        // If the URI contains no value for the Region, bucket or key, the SDK
returns
        // an empty Optional.
        // The SDK returns decoded URI values.

        Region region = s3Uri.region().orElse(null);
        log("region", region);
        // Console output: 'region: us-west-1'.

        String bucket = s3Uri.bucket().orElse(null);
        log("bucket", bucket);
        // Console output: 'bucket: myBucket'.

        String key = s3Uri.key().orElse(null);
        log("key", key);
        // Console output: 'key: resources/doc.txt'.

        Boolean isPathStyle = s3Uri.isPathStyle();
        log("isPathStyle", isPathStyle);
        // Console output: 'isPathStyle: true'.

        // If the URI contains no query parameters, the SDK returns an empty map.
        Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
        log("rawQueryParameters", queryParams);
        // Console output: 'rawQueryParameters: {versionId=[abc123],
partNumber=[77,
        // 88]}'.

        // Retrieve the first or all values for a query parameter as shown in the
        // following code.
        String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
        log("firstMatchingRawQueryParameter-versionId", versionId);
        // Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

        String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
        log("firstMatchingRawQueryParameter-partNumber", partNumber);
        // Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

        List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
```

```
        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters,
 must be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?
query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".",
 the dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
            logger.info("{}: {}", s3UriElement, element);
        }
    }
```

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 オブジェクトのマルチパートコピーを実行する

次のコード例は、Amazon S3 オブジェクトのマルチパートコピーを実行する方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUapiCopyObj
{
    private const string SourceBucket = "doc-example-bucket1";
    private const string TargetBucket = "doc-example-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";

    /// <summary>
    /// This method starts the multi-part upload.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        Console.WriteLine("Copying object...");
        await MPUCopyObjectAsync(s3Client);
    }

    /// <summary>
    /// This method uses the passed client object to perform a multipart
    /// copy operation.
    /// </summary>
    /// <param name="client">An Amazon S3 client object that will be used
```

```
        /// to perform the copy.</param>
        public static async Task MPUCopyObjectAsync(AmazonS3Client client)
        {
            // Create a list to store the copy part responses.
            var copyResponses = new List<CopyPartResponse>();

            // Setup information required to initiate the multipart upload.
            var initiateRequest = new InitiateMultipartUploadRequest
            {
                BucketName = TargetBucket,
                Key = TargetObjectKey,
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            string uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                var metadataRequest = new GetObjectMetadataRequest
                {
                    BucketName = SourceBucket,
                    Key = SourceObjectKey,
                };

                GetObjectMetadataResponse metadataResponse =
                    await client.GetObjectMetadataAsync(metadataRequest);
                var objectSize = metadataResponse.ContentLength; // Length in
bytes.

                // Copy the parts.
                var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    var copyRequest = new CopyPartRequest
                    {
                        DestinationBucket = TargetBucket,
                        DestinationKey = TargetObjectKey,
```

```
                        SourceBucket = SourceBucket,
                        SourceKey = SourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i,
                    };

                    copyResponses.Add(await client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
                }

                // Set up to complete the copy.
                var completeRequest = new CompleteMultipartUploadRequest
                {
                    BucketName = TargetBucket,
                    Key = TargetObjectKey,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error encountered on server.
Message:'{e.Message}' when writing an object");
            }
            catch (Exception e)
            {
                Console.WriteLine($"Unknown encountered on server.
Message:'{e.Message}' when writing an object");
            }
        }
    }
```

- API の詳細については、『AWS SDK for .NET API リファレンス』の以下のトピックを参照してください。
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - GetObjectMetadata
  - UploadPartCopy

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 オブジェクトのマルチパートアップロードを実行する

次のコード例は、Amazon S3 オブジェクトへのマルチパートアップロードを実行する方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

コード例では、次のインポートを使用します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
```

```
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

AWS CRT ベースの S3 クライアント上にある S3 Transfer Manager を使用すると、コンテンツのサイズがしきい値を超えたときにマルチパートアップロードを透過的に実行できます。デフォルトのしきい値は 8 MB です。

```
    public void multipartUploadWithTransferManager(String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b
                        .bucket(bucketName)
                        .key(key))
                .source(Paths.get(filePath))
                .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

S3Client API を使用してマルチパートアップロードを実行します。

```
    public void multipartUploadWithS3Client(String filePath) {

        // Initiate the multipart upload.
```

```
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key));
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            logger.error(e.getMessage());
        }
```

```
        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

マルチパートサポートを有効にして S3AsyncClient API を使用し、マルチパートアップロードを実行します。

```
    public void multipartUploadWithS3AsyncClient(String filePath) {
        // Enable multipart support.
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
                .multipartEnabled(true)
                .build();

        CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b
-> b
                        .bucket(bucketName)
                        .key(key),
                Paths.get(filePath));

        response.join();
        logger.info("File uploaded in multiple 8 MiB parts using
S3AsyncClient.");
    }
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 オブジェクトのアップロードまたはダウンロードを追跡する

次のコード例は、Amazon S3 オブジェクトのアップロードまたはダウンロードを追跡する方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

ファイルのアップロードの進行状況を追跡します。

```
   public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                               String key, URI filePathURI) {
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
               .putObjectRequest(b -> b.bucket(bucketName).key(key))
               .addTransferListener(LoggingTransferListener.create())  // Add
listener.
               .source(Paths.get(filePathURI))
               .build();

       FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

       fileUpload.completionFuture().join();
       /*
           The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
           You can also implement the interface to provide your own logic.

           Configure log4J2 with settings such as the following.
```

```
               <Configuration status="WARN">
                   <Appenders>
                       <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                           <PatternLayout pattern="%m%n"/>
                       </Console>
                   </Appenders>

                   <Loggers>
                       <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                           <AppenderRef ref="AlignedConsoleAppender"/>
                       </logger>
                   </Loggers>
               </Configuration>

           Log4J2 logs the progress. The following is example output for a 21.3
MB file upload.
               Transfer initiated...
               |                    | 0.0%
               |====                | 21.1%
               |============        | 60.5%
               |====================| 100.0%
               Transfer complete!
       */
   }
```

ファイルのダウンロードの進捗状況を追跡します。

```
   public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                             String key, String downloadedFileWithPath) {
       DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
               .getObjectRequest(b -> b.bucket(bucketName).key(key))
               .addTransferListener(LoggingTransferListener.create())  // Add
listener.
               .destination(Paths.get(downloadedFileWithPath))
               .build();

       FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);
```

```
        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        /*
            The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
            You can also implement the interface to provide your own logic.

            Configure log4J2 with settings such as the following.
                <Configuration status="WARN">
                    <Appenders>
                        <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                            <PatternLayout pattern="%m%n"/>
                        </Console>
                    </Appenders>

                    <Loggers>
                        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3
MB file download.
                Transfer initiated...
                |=======             | 39.4%
                |===============     | 78.8%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - GetObject
  - PutObject

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK による単体テストと統合テストのアプローチ例

次のコード例は、AWS SDK を使用して単体テストと統合テストを作成する際のベストプラクティス手法の例を示しています。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

Cargo.toml はテストサンプル用です。

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
  "Doug Schwartz <dougsch@amazon.com>",
]
edition = "2021"

# snippet-start:[testing.rust.Cargo.toml]
[dependencies]
async-trait = "0.1.51"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-credential-types = { version = "1.0.1", features = [ "hardcoded-
credentials", ] }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
aws-types = { version = "1.0.1" }
clap = { version = "~4.4", features = ["derive"] }
```

```
http = "0.2.9"
mockall = "0.11.4"
serde_json = "1"
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
# snippet-end:[testing.rust.Cargo.toml]

[[bin]]
name = "main"
path = "src/main.rs"
```

オートモックとサービスラッパーを使ったユニットテストの例。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// snippet-start:[testing.rust.wrapper]
// snippet-start:[testing.rust.wrapper-uses]
use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};
// snippet-end:[testing.rust.wrapper-uses]

// snippet-start:[testing.rust.wrapper-which-impl]
#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
pub use S3Impl as S3;
// snippet-end:[testing.rust.wrapper-which-impl]

// snippet-start:[testing.rust.wrapper-impl]
#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,
}

#[cfg_attr(test, automock)]
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
```

```
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
    }
}
// snippet-end:[testing.rust.wrapper-impl]

// snippet-start:[testing.rust.wrapper-func]
#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
```

```
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}
// snippet-end:[testing.rust.wrapper-func]
// snippet-end:[testing.rust.wrapper]

// snippet-start:[testing.rust.wrapper-test-mod]
#[cfg(test)]
mod test {
    // snippet-start:[testing.rust.wrapper-tests]
    use super::*;
    use mockall::predicate::eq;

    // snippet-start:[testing.rust.wrapper-test-single]
    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
    }
    // snippet-end:[testing.rust.wrapper-test-single]

    // snippet-start:[testing.rust.wrapper-test-multiple]
    #[tokio::test]
```

```
    async fn test_multiple_pages() {
        // Create the Mock instance with two pages of objects now
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .set_next_continuation_token(Some("next".to_string()))
                    .build())
            });
        mock.expect_list_objects()
            .with(
                eq("test-bucket"),
                eq("test-prefix"),
                eq(Some("next".to_string())),
            )
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(3).build(),
                        s3::types::Object::builder().size(9).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        assert_eq!(19, size);
    }
    // snippet-end:[testing.rust.wrapper-test-multiple]
    // snippet-end:[testing.rust.wrapper-tests]
}
// snippet-end:[testing.rust.wrapper-test-mod]
```

StaticReplayClient を使用した統合テストの例。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// snippet-start:[testing.rust.replay-uses]
use aws_sdk_s3 as s3;
// snippet-end:[testing.rust.replay-uses]

#[allow(dead_code)]
// snippet-start:[testing.rust.replay]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}
// snippet-end:[testing.rust.replay]
```

```
#[allow(dead_code)]
// snippet-start:[testing.rust.replay-tests]
// snippet-start:[testing.rust.replay-make-credentials]
fn make_s3_test_credentials() -> s3::config::Credentials {
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "astestsecretkey",
        Some("atestsessiontoken".to_string()),
        None,
        "",
    )
}
// snippet-end:[testing.rust.replay-make-credentials]

// snippet-start:[testing.rust.replay-test-module]
#[cfg(test)]
mod test {
    // snippet-start:[testing.rust.replay-test-single]
    use super::*;
    use aws_config::BehaviorVersion;
    use aws_sdk_s3 as s3;
    use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
 StaticReplayClient};
    use aws_smithy_types::body::SdkBody;

    #[tokio::test]
    async fn test_single_page() {
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_1.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
```

```
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
        replay_client.assert_requests_match(&[]);
    }
    // snippet-end:[testing.rust.replay-test-single]

    // snippet-start:[testing.rust.replay-test-multiple]
    #[tokio::test]
    async fn test_multiple_pages() {
        // snippet-start:[testing.rust.replay-create-replay]
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
                    .unwrap(),
            );
        let page_2 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
```

```
                    .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
        // snippet-end:[testing.rust.replay-create-replay]
        // snippet-start:[testing.rust.replay-create-client]
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );
        // snippet-end:[testing.rust.replay-create-client]

        // Run the code we want to test with it
        // snippet-start:[testing.rust.replay-test-and-verify]
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        assert_eq!(19, size);

        replay_client.assert_requests_match(&[]);
        // snippet-end:[testing.rust.replay-test-and-verify]
    }
    // snippet-end:[testing.rust.replay-test-multiple]
}
// snippet-end:[testing.rust.replay-tests]
// snippet-end:[testing.rust.replay-test-module]
```

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

ローカルディレクトリを Amazon Simple Storage Service (Amazon S3) バケットに、再帰的にアップロードする

以下のコード例では、ローカルディレクトリを再帰的に Amazon Simple Storage Service (Amazon S3) バケットにアップロードする方法を示します。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

S3TransferManager を使用してローカルディレクトリをアップロードします。完全なファイルとテストを表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public Integer uploadDirectory(S3TransferManager transferManager,
            URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
 transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());
```

```
        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「UploadDirectory」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 との間で大きなファイルをアップロードまたはダウンロードする

次のコード例は、Amazon S3 との間で大きなファイルをアップロードまたはダウンロードする方法を示しています。

詳細については、「マルチパートアップロードを使用したオブジェクトのアップロード」を参照してください。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

Amazon S3 TransferUtility を使用して、S3 バケットとの間でファイルを転送する関数を呼び出します。

```
global using System.Text;
global using Amazon.S3;
```

```
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;



// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
 $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
\TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");

var success = await TransferMethods.UploadSingleFileAsync(transferUtil,
 bucketName, fileToUpload, localPath);
if (success)
```

```
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
 {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil,
 bucketName, keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
 {bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];

Console.WriteLine($"Downloading {keyName} from {bucketName}.");

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
 keyName, localPath);
if (success)
```

```
{
    Console.WriteLine("$Successfully downloaded the file, {keyName} from
 {bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil,
 bucketName, s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to
 {downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}

// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);
```

```
    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
 Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to
 an\n\t  S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
 directory.");
    Console.WriteLine($"\n{sepBar}");
}

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth -
 textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
```

```
    }
}

// Displays a list of the files in the specified S3 bucket and prefix.
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
 s3Path)
{
    ListObjectsV2Request request = new()
    {
        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}
```

1 つのファイルをアップロードします。

```
        /// <summary>
        /// Uploads a single file from the local computer to an S3 bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the file
        /// will be stored.</param>
        /// <param name="fileName">The name of the file to upload.</param>
```

```
        /// <param name="localPath">The local path where the file is stored.</
param>
        /// <returns>A boolean value indicating the success of the action.</
returns>
        public static async Task<bool> UploadSingleFileAsync(
            TransferUtility transferUtil,
            string bucketName,
            string fileName,
            string localPath)
        {
            if (File.Exists($"{localPath}\\{fileName}"))
            {
                try
                {
                    await transferUtil.UploadAsync(new
 TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        Key = fileName,
                        FilePath = $"{localPath}\\{fileName}",
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Could not upload {fileName} from
 {localPath} because:");
                    Console.WriteLine(s3Ex.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"{fileName} does not exist in {localPath}");
                return false;
            }
        }
```

ローカルディレクトリ全体をアップロードします。

```
        /// <summary>
        /// Uploads all the files in a local directory to a directory in an S3
        /// bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the files
        /// will be stored.</param>
        /// <param name="keyPrefix">The key prefix is the S3 directory where
        /// the files will be stored.</param>
        /// <param name="localPath">The local directory that contains the files
        /// to be uploaded.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> UploadFullDirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string keyPrefix,
            string localPath)
        {
            if (Directory.Exists(localPath))
            {
                try
                {
                    await transferUtil.UploadDirectoryAsync(new
 TransferUtilityUploadDirectoryRequest
                    {
                        BucketName = bucketName,
                        KeyPrefix = keyPrefix,
                        Directory = localPath,
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Can't upload the contents of {localPath}
 because:");
                    Console.WriteLine(s3Ex?.Message);
                    return false;
                }
            }
            else
            {
```

```
                Console.WriteLine($"The directory {localPath} does not exist.");
                return false;
            }
        }
```

1 つのファイルをダウンロードします。

```
        /// <summary>
        /// Download a single file from an S3 bucket to the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// file to download.</param>
        /// <param name="keyName">The name of the file to download.</param>
        /// <param name="localPath">The path on the local computer where the
        /// downloaded file will be saved.</param>
        /// <returns>A Boolean value indicating the results of the action.</
returns>
        public static async Task<bool> DownloadSingleFileAsync(
        TransferUtility transferUtil,
            string bucketName,
            string keyName,
            string localPath)
        {
            await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
            {
                BucketName = bucketName,
                Key = keyName,
                FilePath = $"{localPath}\\{keyName}",
            });

            return (File.Exists($"{localPath}\\{keyName}"));
        }
```

S3 バケットの内容をダウンロードします。

```
        /// <summary>
        /// Downloads the contents of a directory in an S3 bucket to a
        /// directory on the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The bucket containing the files to
 download.</param>
        /// <param name="s3Path">The S3 directory where the files are located.</
param>
        /// <param name="localPath">The local path to which the files will be
        /// saved.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> DownloadS3DirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string s3Path,
            string localPath)
        {
            int fileCount = 0;

            // If the directory doesn't exist, it will be created.
            if (Directory.Exists(s3Path))
            {
                var files = Directory.GetFiles(localPath);
                fileCount = files.Length;
            }

            await transferUtil.DownloadDirectoryAsync(new
 TransferUtilityDownloadDirectoryRequest
            {
                BucketName = bucketName,
                LocalDirectory = localPath,
                S3Directory = s3Path,
            });

            if (Directory.Exists(localPath))
            {
                var files = Directory.GetFiles(localPath);
                if (files.Length > fileCount)
                {
                    return true;
                }
```

```
            // No change in the number of files. Assume
            // the download failed.
            return false;
        }

        // The local directory doesn't exist. No files
        // were downloaded.
        return false;
    }
```

TransferUtility を使用してアップロードの進行状況を追跡します。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;

/// <summary>
/// This example shows how to track the progress of a multipart upload
/// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
/// upload to an Amazon S3 bucket.
/// </summary>
public class TrackMPUUsingHighLevelAPI
{
    public static async Task Main()
    {
        string bucketName = "doc-example-bucket";
        string keyName = "sample_pic.png";
        string path = "filepath/directory/";
        string filePath = $"{path}{keyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();

        await TrackMPUAsync(client, bucketName, filePath, keyName);
    }
```

```
        /// <summary>
        /// Starts an Amazon S3 multipart upload and assigns an event handler to
        /// track the progress of the upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// perform the multipart upload.</param>
        /// <param name="bucketName">The name of the bucket to which to upload
        /// the file.</param>
        /// <param name="filePath">The path, including the file name of the
        /// file to be uploaded to the Amazon S3 bucket.</param>
        /// <param name="keyName">The file name to be used in the
        /// destination Amazon S3 bucket.</param>
        public static async Task TrackMPUAsync(
            IAmazonS3 client,
            string bucketName,
            string filePath,
            string keyName)
        {
            try
            {
                var fileTransferUtility = new TransferUtility(client);

                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName,
                    };

                uploadRequest.UploadProgressEvent +=
                    new EventHandler<UploadProgressArgs>(
                        UploadRequest_UploadPartProgressEvent);

                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:: {ex.Message}");
            }
        }
```

```
        /// <summary>
        /// Event handler to check the progress of the multipart upload.
        /// </summary>
        /// <param name="sender">The object that raised the event.</param>
        /// <param name="e">The object that contains multipart upload
        /// information.</param>
        public static void UploadRequest_UploadPartProgressEvent(object sender,
UploadProgressArgs e)
        {
            // Process event.
            Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
        }
    }
```

暗号化したオブジェクトをアップロードします。

```
    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Security.Cryptography;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
    /// perform a multipart upload to an Amazon S3 bucket.
    /// </summary>
    public class SSECLowLevelMPUcopyObject
    {
        public static async Task Main()
        {
            string existingBucketName = "doc-example-bucket";
            string sourceKeyName = "sample_file.txt";
            string targetKeyName = "sample_file_copy.txt";
            string filePath = $"sample\\{targetKeyName}";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
```

```
            // For example: RegionEndpoint.USEast1.
            IAmazonS3 client = new AmazonS3Client();

            // Create the encryption key.
            var base64Key = CreateEncryptionKey();

            await CreateSampleObjUsingClientEncryptionKeyAsync(
                client,
                existingBucketName,
                sourceKeyName,
                filePath,
                base64Key);
        }

        /// <summary>
        /// Creates the encryption key to use with the multipart upload.
        /// </summary>
        /// <returns>A string containing the base64-encoded key for encrypting
        /// the multipart upload.</returns>
        public static string CreateEncryptionKey()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);
            return base64Key;
        }

        /// <summary>
        /// Creates and uploads an object using a multipart upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object used to
        /// initialize and perform the multipart upload.</param>
        /// <param name="existingBucketName">The name of the bucket to which
        /// the object will be uploaded.</param>
        /// <param name="sourceKeyName">The source object name.</param>
        /// <param name="filePath">The location of the source object.</param>
        /// <param name="base64Key">The encryption key to use with the upload.</
param>
        public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
            IAmazonS3 client,
            string existingBucketName,
            string sourceKeyName,
            string filePath,
```

```
            string base64Key)
        {
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
               await client.InitiateMultipartUploadAsync(initiateRequest);

            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
                        PartNumber = i,
                        PartSize = partSize,
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key,
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));
```

```
                filePosition += partSize;
            }

            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                UploadId = initResponse.UploadId,
            };
            completeRequest.AddPartETags(uploadResponses);

            CompleteMultipartUploadResponse completeUploadResponse =
                await client.CompleteMultipartUploadAsync(completeRequest);
        }
        catch (Exception exception)
        {
            Console.WriteLine($"Exception occurred: {exception.Message}");

            // If there was an error, abort the multipart upload.
            AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                UploadId = initResponse.UploadId,
            };

            await client.AbortMultipartUploadAsync(abortMPURequest);
        }
    }
}
```

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

アップロードマネージャーを使用してデータを分割し、同時にアップロードすることで、大きなオブジェクトをアップロードすることができます。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadLargeObject uses an upload manager to upload data to an object in a
 bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(bucketName string, objectKey string,
 largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
```

```
 if err != nil {
  log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }

 return err
}
```

ダウンロードマネージャーを使用して、データを分割して取得し、同時にダウンロードすることで、大きなオブジェクトをダウンロードすることができます。

```
// DownloadLargeObject uses a download manager to download an object from a
 bucket.
// The download manager gets the data in parts and writes them to a buffer until
 all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(bucketName string, objectKey
 string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
 {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(context.TODO(), buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

S3TransferManager を使用して、S3 バケットとの間でファイルを転送する関数を呼び出します。

```
   public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
           URI destinationPathURI, String bucketName) {
       DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
               .destination(Paths.get(destinationPathURI))
               .bucket(bucketName)
               .build());
       CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

       completedDirectoryDownload.failedTransfers()
               .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
       return completedDirectoryDownload.failedTransfers().size();
   }
```

ローカルディレクトリ全体をアップロードします。

```
   public Integer uploadDirectory(S3TransferManager transferManager,
           URI sourceDirectory, String bucketName) {
       DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
               .source(Paths.get(sourceDirectory))
               .bucket(bucketName)
               .build());

       CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
```

```
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

1 つのファイルをアップロードします。

```
    public String uploadFile(S3TransferManager transferManager, String
bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

サイズの大きいファイルをアップロードします。

```
import {
  CreateMultipartUploadCommand,
  UploadPartCommand,
  CompleteMultipartUploadCommand,
  AbortMultipartUploadCommand,
```

```
  S3Client,
} from "@aws-sdk/client-s3";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

export const main = async () => {
  const s3Client = new S3Client({});
  const bucketName = "test-bucket";
  const key = "multipart.txt";
  const str = createString();
  const buffer = Buffer.from(str, "utf8");

  let uploadId;

  try {
    const multipartUpload = await s3Client.send(
      new CreateMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );

    uploadId = multipartUpload.UploadId;

    const uploadPromises = [];
    // Multipart uploads require a minimum size of 5 MB per part.
    const partSize = Math.ceil(buffer.length / 5);

    // Upload each part.
    for (let i = 0; i < 5; i++) {
      const start = i * partSize;
      const end = start + partSize;
      uploadPromises.push(
        s3Client
          .send(
            new UploadPartCommand({
              Bucket: bucketName,
              Key: key,
              UploadId: uploadId,
              Body: buffer.subarray(start, end),
```

```
              PartNumber: i + 1,
            }),
          )
          .then((d) => {
            console.log("Part", i + 1, "uploaded");
            return d;
          }),
      );
    }

    const uploadResults = await Promise.all(uploadPromises);

    return await s3Client.send(
      new CompleteMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
        MultipartUpload: {
          Parts: uploadResults.map(({ ETag }, i) => ({
            ETag,
            PartNumber: i + 1,
          })),
        },
      }),
    );

    // Verify the output by downloading the file from the Amazon Simple Storage
 Service (Amazon S3) console.
    // Because the output is a 25 MB string, text editors might struggle to open
 the file.
  } catch (err) {
    console.error(err);

    if (uploadId) {
      const abortCommand = new AbortMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
      });

      await s3Client.send(abortCommand);
    }
  }
};
```

サイズの大きいファイルをダウンロードします。

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream } from "fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: parseInt(start),
    end: parseInt(end),
    length: parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

// When downloading a large file, you might want to break it down into
// smaller pieces. Amazon S3 accepts a Range header to specify the start
// and end of the byte range to be downloaded.
const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));
```

```
  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    console.log(`Downloading bytes ${nextRange.start} to ${nextRange.end}`);

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
};

export const main = async () => {
  await downloadInChunks({
    bucket: "my-cool-bucket",
    key: "my-cool-object.txt",
  });
};
```

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

使用可能な転送マネージャの設定を使用して、ファイルを転送する関数を作成します。コールバッククラスを使用して、ファイル転送中にコールバックの進行状況を書き込みます。

```
import sys
```

```
import threading

import boto3
from boto3.s3.transfer import TransferConfig


MB = 1024 * 1024
s3 = boto3.resource("s3")


class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
```

```
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
```

```
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
 sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

```
def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
```

```
            download_file_path, Config=config, Callback=transfer_callback
        )
        return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
 sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

転送マネージャの機能とレポート結果のデモンストレーションを行います。

```
import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
```

```
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")


class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
 files? "
            )
```

```
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
                "demonstration files? "
            )
            try:
                self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
            except ParamValidationError as err:
                print(err)
                self.demo_bucket = None
            except ClientError as err:
                print(err)
                print(
                    f"Either {self.demo_bucket} doesn't exist or you don't "
                    f"have access to it."
                )
                self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.

        Ask the user if they want to run this specific demonstration.
        If they say yes, create a file on the local path, upload it
        using the specified upload function, then download it using the
        specified download function.
        """
        if download_args is None:
            download_args = {}
        if upload_args is None:
            upload_args = {}
        question = question.format(self.file_size_mb)
        answer = input(f"{question} (y/n)")
        if answer.lower() == "y":
            local_file_path, object_key, download_file_path =
self._create_demo_file()

            file_transfer.TransferConfig = self._config_wrapper(
```

```
                TransferConfig, CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Uploading", local_file_path, object_key, **upload_args
            )
            start_time = time.perf_counter()
            thread_info = upload_func(
                local_file_path,
                self.demo_bucket,
                object_key,
                self.file_size_mb,
                **upload_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Downloading", object_key, download_file_path, **download_args
            )
            start_time = time.perf_counter()
            thread_info = download_func(
                self.demo_bucket,
                object_key,
                download_file_path,
                self.file_size_mb,
                **download_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

    def last_name_set(self):
        """Get the name set used for the last demo."""
        return self._chore_list[-1]

    def cleanup(self):
        """
        Remove files from the demo folder, and uploaded objects from the
        Amazon S3 bucket.
        """
        print("-" * self._terminal_width)
```

```
        for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
            print(f"Removing {local_file_path}")
            try:
                os.remove(local_file_path)
            except FileNotFoundError as err:
                print(err)

            print(f"Removing {downloaded_file_path}")
            try:
                os.remove(downloaded_file_path)
            except FileNotFoundError as err:
                print(err)

            if self.demo_bucket:
                print(f"Removing {self.demo_bucket}:{s3_object_key}")
                try:

self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
                except ClientError as err:
                    print(err)

    def _setup_platform_specific(self):
        """Set up platform-specific command used to create a large file."""
        if platform.system() == "Windows":
            self._create_file_cmd = "fsutil file createnew {} {}"
            self._size_multiplier = MB
        elif platform.system() == "Linux" or platform.system() == "Darwin":
            self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={MB}
count={{}}"
            self._size_multiplier = 1
        else:
            raise EnvironmentError(
                f"Demo of platform {platform.system()} isn't supported."
            )

    def _create_demo_file(self):
        """
        Create a file in the demo folder specified by the user. Store the local
        path, object name, and download path for later cleanup.

        Only the local file is created by this method. The Amazon S3 object and
        download file are created later during the demonstration.
```

```
        Returns:
        A tuple that contains the local file path, object name, and download
        file path.
        """
        file_name_template = "TestFile{}-{}.demo"
        local_suffix = "local"
        object_suffix = "s3object"
        download_suffix = "downloaded"
        file_tag = len(self._chore_list) + 1

        local_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag, local_suffix)
        )

        s3_object_key = file_name_template.format(file_tag, object_suffix)

        downloaded_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag,
download_suffix)
        )

        filled_cmd = self._create_file_cmd.format(
            local_file_path, self.file_size_mb * self._size_multiplier
        )

        print(
            f"Creating file of size {self.file_size_mb} MB "
            f"in {self.demo_folder} by running:"
        )
        print(f"{'':4}{filled_cmd}")
        os.system(filled_cmd)

        chore = (local_file_path, s3_object_key, downloaded_file_path)
        self._chore_list.append(chore)
        return chore

    def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
        """Report configuration and extra arguments used for a file transfer."""
        print("-" * self._terminal_width)
        print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
        if kwargs:
            print("With extra args:")
            for arg, value in kwargs.items():
                print(f'{"":4}{arg:<20}: {value}')
```

```
    @staticmethod
    def ask_user(question):
        """
        Ask the user a yes or no question.

        Returns:
        True when the user answers 'y' or 'Y'; otherwise, False.
        """
        answer = input(f"{question} (y/n) ")
        return answer.lower() == "y"

    @staticmethod
    def _config_wrapper(func, config_attrs):
        def wrapper(*args, **kwargs):
            config = func(*args, **kwargs)
            print("With configuration:")
            for attr in config_attrs:
                print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
            return config

        return wrapper

    @staticmethod
    def _report_transfer_result(thread_info, elapsed):
        """Report the result of a transfer, including per-thread data."""
        print(f"\nUsed {len(thread_info)} threads.")
        for ident, byte_count in thread_info.items():
            print(f"{'':4}Thread {ident} copied {byte_count} bytes.")
        print(f"Your transfer took {elapsed:.2f} seconds.")


def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
```

```
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
        },
    )

    # Upload using server-side encryption with customer-provided
    # encryption keys.
    # Generate a 256-bit key from a passphrase.
    sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
    demo_manager.demo(
        "Do you want to upload and download a {} MB file using "
        "server-side encryption?",
        file_transfer.upload_with_sse,
        file_transfer.download_with_sse,
        upload_args={"sse_key": sse_key},
        download_args={"sse_key": sse_key},
    )
```

```
    # Download without specifying an encryption key to show that the
    # encryption key must be included to download an encrypted object.
    if demo_manager.ask_user(
        "Do you want to try to download the encrypted "
        "object without sending the required key?"
    ):
        try:
            _, object_key, download_file_path = demo_manager.last_name_set()
            file_transfer.download_with_default_configuration(
                demo_manager.demo_bucket,
                object_key,
                download_file_path,
                demo_manager.file_size_mb,
            )
        except ClientError as err:
            print(
                "Got expected error when trying to download an encrypted "
                "object without specifying encryption info:"
            )
            print(f"{'':4}{err}")

    # Remove all created and downloaded files, remove all objects from
    # S3 storage.
    if demo_manager.ask_user(
        "Demonstration complete. Do you want to remove local files " "and S3
 objects?"
    ):
        demo_manager.cleanup()


if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
 get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_service::error::Error;
use std::process;
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}
```

```
async fn run_example() -> Result<(), Error> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("doc-example-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    s3_service::create_bucket(&client, &bucket_name, region.as_ref()).await?;

    let key = "sample.txt".to_string();
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await
        .unwrap();
    let upload_id = multipart_upload_res.upload_id().unwrap();

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)
            .take(256)
            .map(char::from)
            .collect();
        let return_string: String = "\n".to_string();
        file.write_all(rand_string.as_ref())
            .expect("Error writing to file.");
        file.write_all(return_string.as_ref())
            .expect("Error writing to file.");
    }

    let path = Path::new(&key);
    let file_size = tokio::fs::metadata(path)
        .await
        .expect("it exists I swear")
        .len();

    let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
    let mut size_of_last_chunk = file_size % CHUNK_SIZE;
```

```
    if size_of_last_chunk == 0 {
        size_of_last_chunk = CHUNK_SIZE;
        chunk_count -= 1;
    }

    if file_size == 0 {
        panic!("Bad file size.");
    }
    if chunk_count > MAX_CHUNKS {
        panic!("Too many chunks! Try increasing your chunk size.")
    }

    let mut upload_parts: Vec<CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();
        //Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;
        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
```

```
    }
    let completed_multipart_upload: CompletedMultipartUpload =
 CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await
        .unwrap();

    let data: GetObjectOutput = s3_service::download_object(&client,
 &bucket_name, &key).await?;
    let data_length: u64 = data
        .content_length()
        .unwrap_or_default()
        .try_into()
        .unwrap();
    if file.metadata().unwrap().len() == data_length {
        println!("Data lengths match.");
    } else {
        println!("The data was not the same size!");
    }

    s3_service::delete_objects(&client, &bucket_name)
        .await
        .expect("Error emptying bucket.");
    s3_service::delete_bucket(&client, &bucket_name)
        .await
        .expect("Error deleting bucket.");

    Ok(())
}
```

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

サイズが不明なストリームを AWS SDK を使用して Amazon S3 オブジェクトにアップロードする

次のコード例は、サイズが不明なストリームを Amazon S3 オブジェクトにアップロードする方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

AWS CRT ベースの S3 クライアントを使用します。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

    /**
     * @param s33CrtAsyncClient - To upload content from a stream of unknown
 size, use the AWS CRT-based S3 client. For more information, see
     *                          https://docs.aws.amazon.com/sdk-for-java/latest/
developer-guide/crt-based-s3-client.html.
     * @param bucketName - The name of the bucket.
```

```
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
 Returns metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStream(S3AsyncClient s33CrtAsyncClient,
 String bucketName, String key) {

        BlockingInputStreamAsyncRequestBody body =
                AsyncRequestBody.forBlockingInputStream(null); // 'null'
 indicates a stream will be provided later.

        CompletableFuture<PutObjectResponse> responseFuture =
                s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
 body);

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());

        // Provide the stream of data to be uploaded.
        body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        return response;
    }
}
```

Amazon S3 Transfer Manager を使用します。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;
```

```
import java.io.ByteArrayInputStream;
import java.util.UUID;

    /**
     * @param transferManager - To upload content from a stream of unknown size,
 use the S3TransferManager based on the AWS CRT-based S3 client.
     *                       For more information, see https://
docs.aws.amazon.com/sdk-for-java/latest/developer-guide/transfer-manager.html.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
 result of the completed upload.
     */
    public CompletedUpload uploadStream(S3TransferManager transferManager, String
 bucketName, String key) {

        BlockingInputStreamAsyncRequestBody body =
                AsyncRequestBody.forBlockingInputStream(null); // 'null'
 indicates a stream will be provided later.

        Upload upload = transferManager.upload(builder -> builder
                .requestBody(body)
                .putObjectRequest(req -> req.bucket(bucketName).key(key))
                .build());

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());

        // Provide the stream of data to be uploaded.
        body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

        return upload.completionFuture().join();
    }
}
```

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

チェックサムを使用した AWS SDK での Amazon S3 オブジェクトの操作

次のコード例は、チェックサムを使用して Amazon S3 オブジェクトを操作する方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

コード例では、以下のインポートのサブセットを使用しています。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
```

```
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

PutObjectRequest をビルドするときに、putObject メソッドのチェックサムアルゴリズムを指定します。

```
    public void putObjectWithChecksum() {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.CRC32),
                RequestBody.fromString("This is a test"));
    }
```

GetObjectRequest をビルドするときに、getObject メソッドのチェックサムを確認します。

```
    public GetObjectResponse getObjectWithChecksum() {
        return s3Client.getObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumMode(ChecksumMode.ENABLED))
                .response();
    }
```

PutObjectRequest をビルドするときに、putObject メソッドのチェックサムを事前に計算します。

```
    public void putObjectWithPrecalculatedChecksum(String filePath) {
        String checksum = calculateChecksum(filePath, "SHA-256");

        s3Client.putObject((b -> b
                .bucket(bucketName)
                .key(key)
                .checksumSHA256(checksum)),
                RequestBody.fromFile(Paths.get(filePath)));
    }
```

AWS CRT ベースの S3 クライアント上にある S3 Transfer Manager を使用して、コンテンツのサイズがしきい値を超えたときにマルチパートアップロードを透過的に実行します。デフォルトのしきい値は 8 MB です。

SDK で使用するチェックサムアルゴリズムを指定できます。デフォルトでは、SDK は CRC32 アルゴリズムを使用します。

```
    public void multipartUploadWithChecksumTm(String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b
                        .bucket(bucketName)
                        .key(key)
                        .checksumAlgorithm(ChecksumAlgorithm.SHA1))
                .source(Paths.get(filePath))
                .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

S3Client API または (S3AsyncClient API) を使用してマルチパートアップロードを実行します。追加のチェックサムを指定する場合は、アップロードの開始時に使用するアルゴリズムを指定する必要があります。また、パートリクエストごとにアルゴリズムを指定し、アップロード後にパートごとに計算されたチェックサムを指定する必要があります。

```
    public void multipartUploadWithChecksumS3Client(String filePath) {
        ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;
```

```
        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(algorithm)); // Checksum specified on
initiation.
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .checksumAlgorithm(algorithm) // Checksum specified on
each part.
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .checksumCRC32(partResponse.checksumCRC32()) // Provide
the calculated checksum.
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);
```

```
                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して、Amazon S3 のバージョン管理されたオブジェクトを操作する

次のコードサンプルは、以下の操作方法を示しています。

- バージョニングされた S3 バケットを作成します。
- オブジェクトのすべてのバージョンを取得します。
- オブジェクトを以前のバージョンにロールバックします。
- バージョニングされたオブジェクトを削除して復元します。
- オブジェクトのバージョンを完全に削除します。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SS3 アクションをラップする関数を作成します。

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
```

```
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
 expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket



def rollback_object(bucket, object_key, version_id):
    """
```

```
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest
 version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
```

```
        logger.warning("Got an active version for %s, nothing to do.",
 object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)



def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

詩のスタンザをバージョニングされたオブジェクトにアップロードし、一連のアクションを実行します。

```
def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
```

```
    print(
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
    print("-" * width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
    obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
    print(
        "The latest version of the stanza is now:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Versions are returned in order, most recent first.
    obj_stanza_1_versions =
 bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    print(
        "The version data of the stanza revisions:",
        *[
            f"    {version.version_id}, last modified {version.last_modified}"
            for version in obj_stanza_1_versions
        ],
        sep="\n",
    )

    # Rollback two versions.
    print("\nRolling back two versions...")
    rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
    print(
        "The latest version of the stanza:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
```

```
    )

    # Delete the stanza
    print("\nDeleting the stanza...")
    obj_stanza_1.delete()
    try:
        obj_stanza_1.get()
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            print("The stanza is now deleted (as expected).")
        else:
            raise

    # Revive the stanza
    print("\nRestoring the stanza...")
    revive_object(bucket, obj_stanza_1.key)
    print(
        "The stanza is restored! The latest version is again:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Permanently delete all versions of the object. This cannot be undone!
    print("\nPermanently deleting all versions of the stanza...")
    permanently_delete_object(bucket, obj_stanza_1.key)
    obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    if len(list(obj_stanza_1_versions)) == 0:
        print("The stanza has been permanently deleted and now has no versions.")
    else:
        print("Something went wrong. The stanza still exists!")

    print(f"\nRemoving {bucket.name}...")
    bucket.delete()
    print(f"{bucket.name} deleted.")
    print("Demo done!")
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - CreateBucket

- DeleteObject
- ListObjectVersions
- PutBucketLifecycleConfiguration

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用した Amazon S3 用のサーバーレスサンプル

次のコード例は、AWS SDK で Amazon S3 を使用する方法を示します。

例

- Amazon S3 トリガーから Lambda 関数を呼び出す

Amazon S3 トリガーから Lambda 関数を呼び出す

次のコード例は、S3 バケットにオブジェクトをアップロードすることによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数は、イベントパラメータから S3 バケット名とオブジェクトキーを取得し、Amazon S3 API を呼び出してオブジェクトのコンテンツタイプを取得してログに記録します。

.NET

AWS SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

.NET を使用して Lambda で S3 イベントを消費します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
```

```
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }

        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }

        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
            {
                if (evt.Records.Count <= 0)
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");

                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");

                return objectResult.Key;
```

```
            }
            catch (Exception e)
            {
                context.Logger.LogLine($"Error processing request -
 {e.Message}");

                return string.Empty;
            }
        }
    }
}
```

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Go を使用して Lambda で S3 イベントを消費します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Printf("failed to load default config: %s", err)
```

```
  return err
 }
 s3Client := s3.NewFromConfig(sdkConfig)

 for _, record := range s3Event.Records {
  bucket := record.S3.Bucket.Name
  key := record.S3.Object.URLDecodedKey
  headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
   Bucket: &bucket,
   Key:    &key,
  })
  if err != nil {
   log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
   return err
  }
  log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
 *headOutput.ContentType)
 }

 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用して Lambda で S3 イベントを消費します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;
```

```
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
          HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);

          logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

          return "Ok";
        } catch (Exception e) {
          throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
```

```
}
```

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

JavaScript を使用して Lambda で S3 イベントを消費します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

exports.handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g,
 ' '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make
 sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
```

```
    }
};
```

TypeScript を使用して Lambda で S3 イベントを消費する。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
    return ContentType;
  } catch (err) {
    console.log(err);
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure
 they exist and your bucket is in the same region as this function.`;
    console.log(message);
    throw new Error(message);
  }
};
```

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

PHP を使用して Lambda で S3 イベントの消費。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());
```

```
            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
 $bucket . '. Make sure they exist and your bucket is in the same region as this
 function.' . "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Python を使用して Lambda で S3 イベントを消費します。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')
```

```
def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
 encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and
 your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用して Lambda での S3 イベントの消費。

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
```

```
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Rust を使用して Lambda で S3 イベントを消費します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};


/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
```

```
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
 SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
 name to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
 exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;

    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
 contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }
```

```
    Ok(())
}
```

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用した Amazon S3 のクロスサービスの例

次のサンプルアプリケーションでは、AWS SDK を使用して Amazon S3 を他の AWS のサービスと組み合わせます。それぞれの例には、GitHub へのリンクがあり、アプリケーションを設定および実行する方法についての説明を参照できます。

例

- Amazon Transcribe アプリを構築する
- AWS SDK を使用して、テキストを音声に変換し、テキストに戻す
- ユーザーがラベルを使用して写真を管理できる写真アセット管理アプリケーションの作成
- Amazon Textract エクスプローラーアプリケーションを作成する
- AWS SDK を使用して、Amazon Rekognition でイメージ内の PPE を検出する
- AWS SDK を使用して、画像から抽出されたテキスト内のエンティティを検出する
- AWS SDK を使用してイメージ内の顔を検出する
- AWS SDK を使用して、Amazon Rekognition で、イメージ内のオブジェクトを検出します
- AWS SDK を使用して、Amazon Rekognition で、動画内の人やオブジェクトを検出する
- AWS SDK を使用して、EXIF およびその他のイメージ情報を保存します
- S3 Object Lambda でアプリケーションのデータを変換する

Amazon Transcribe アプリを構築する

次のコード例は、Amazon Transcribe を使用して音声録音を文字起こしし、ブラウザに表示する方法を示しています。

JavaScript

SDK for JavaScript (v3)

Amazon Transcribe を使用して音声録音を文字起こしし、ブラウザに表示するアプリを作成します。このアプリは 2 つの Amazon Simple Storage Service (Amazon S3) バケットを使用します。1 つはアプリケーションコードをホストし、もう 1 つは文字起こしを保存します。このアプリは、Amazon Cognito ユーザープールを使用してユーザーを認証します。認証済みユーザーは、必要な AWS サービスにアクセスする AWS Identity and Access Management (IAM) 許可があります。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例は、AWS SDK for JavaScript v3 デベロッパーガイドでも使用できます。

この例で使用されているサービス

- Amazon Cognito ID
- Amazon S3
- Amazon Transcribe

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して、テキストを音声に変換し、テキストに戻す

次のコードサンプルは、以下の操作方法を示しています。

- Amazon Polly を使用して、プレーンテキスト (UTF-8) 入力ファイルを音声ファイルに合成します。
- Amazon S3 バケットに音声ファイルをアップロードします。
- Amazon Transcribe を使用して、音声ファイルをテキストに変換します。
- テキストを表示します。

Rust

SDK for Rust

Amazon Polly を使用して、プレーンテキスト (UTF-8) 入力ファイルを音声ファイルに合成し、音声ファイルを Amazon S3 バケットにアップロードし、Amazon Transcribe を使用して音声ファイルをテキストに変換し、テキストを表示します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Polly
- Amazon S3
- Amazon Transcribe

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

ユーザーがラベルを使用して写真を管理できる写真アセット管理アプリケーションの作成

以下のコード例は、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションの作成方法を示しています。

.NET

AWS SDK for .NET

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK for C++

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK for Java 2.x

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK for JavaScript (v3)

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK for Kotlin

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK for PHP

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK for Rust

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon Textract エクスプローラーアプリケーションを作成する

次のコード例は、インタラクティブアプリケーションで Amazon Textract の出力を調べる方法を示しています。

JavaScript

SDK for JavaScript (v3)

AWS SDK for JavaScript を使用して、Amazon Textract を使用してドキュメントイメージからデータを抽出し、インタラクティブなウェブページに表示する React アプリケーションを構築する方法を示します。この例はウェブブラウザで実行され、認証情報に認証された Amazon Cognito ID が必要です。Amazon Simple Storage Service (Amazon S3) をストレージに使用し、通知のために、Amazon Simple Notification Service (Amazon SNS) トピックにサブスクライブした Amazon Simple Queue Service (Amazon SQS) キューをポーリングします。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Cognito ID
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK for Python (Boto3)

Amazon Textract で AWS SDK for Python (Boto3) を使用して、ドキュメントイメージ内のテキスト、フォーム、および表要素を検出する方法を示します。入力イメージと Amazon Textract 出力は、検出された要素を探索できる Tkinter アプリケーションに表示されます。

- Amazon Textract にドキュメントイメージを送信し、検出された要素の出力を調べます。
- Amazon Textract に直接イメージを送信するか、Amazon Simple Storage Service (Amazon S3) バケットを通じてイメージを送信します。
- 非同期 API を使用して、ジョブの完了時に Amazon Simple Notification Service (Amazon SNS) トピックに通知を発行するジョブを開始します。
- Amazon Simple Queue Service (Amazon SQS) キューにジョブ完了メッセージについてポーリングし、結果を表示します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して、Amazon Rekognition でイメージ内の PPE を検出する

次のコード例は、Amazon Rekognition を使用して画像内の個人用防護具 (PPE) を検出するアプリケーションを構築する方法を示しています。

Java

SDK for Java 2.x

AWS Lambda 個人用保護具のイメージを検出する関数の作成方法を示します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK for JavaScript (v3)

Amazon Rekognition で、AWS SDK for JavaScript を使用して Amazon Simple Storage Service (Amazon S3) バケット内にあるイメージで個人用保護具 (PPE) を検出するアプリケーションを作成する方法を示します。アプリケーションは、結果を Amazon DynamoDB テーブルに保存し、Amazon Simple Email Service (Amazon SES) を使用して結果を記載した E メール通知を管理者に送信します。

以下ではその方法を説明しています。

- Amazon Cognito を使用して認証されていないユーザーを作成します。
- Amazon Rekognition を使用して、PPE の画像を分析します。
- Amazon SES の E メールアドレスを検証します。
- 結果で DynamoDB テーブルを更新します。

- Amazon SES を使用して E メール通知を送信します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して、画像から抽出されたテキスト内のエンティティを検出する

次のコード例は、Amazon Comprehend を使用して、Amazon S3 に格納されている画像から Amazon Textract によって抽出されたテキスト内のエンティティを検出する方法を示しています。

Python

SDK for Python (Boto3)

Jupyter Notebook で AWS SDK for Python (Boto3) を使用して、画像から抽出されたテキスト内のエンティティを検出する方法を示しています。この例では、Amazon Textract を使用して Amazon Simple Storage Service (Amazon S3) に保存されている画像からテキストを抽出し、Amazon Comprehend を使用して、抽出されたテキスト内のエンティティを検出します。

この例は Jupyter Notebook であり、ノートブックをホストできる環境で実行する必要があります。Amazon SageMaker を使用してサンプルを実行する方法については、「TextTractAndComprehendNotebook.ipynb」の手順を参照してください。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Comprehend
- Amazon S3
- Amazon Textract

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用してイメージ内の顔を検出する

次のコードサンプルは、以下の操作方法を示しています。

- イメージを Amazon S3 バケットに保存します。
- Amazon Rekognition を使用して、年齢範囲、性別、感情 (笑顔など) などの顔の詳細を検出します。
- これらの詳細を表示します。

Rust

SDK for Rust

アップロード プレフィックスを付け、Amazon S3 バケット内でイメージを保存し、Amazon Rekognition を使用して、年齢層、性別、感情 (笑顔など) などの顔の詳細を検出し、それらの詳細を表示します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して、Amazon Rekognition で、イメージ内のオブジェクトを検出します

次のコード例は、Amazon Rekognition を使用して画像内からカテゴリ別にオブジェクトを検出するアプリケーションを構築する方法を示しています。

.NET

AWS SDK for .NET

Amazon Simple Storage Service (Amazon S3) バケットにあるイメージ内から、Amazon Rekognition を使用してカテゴリ別にオブジェクトを識別するアプリケーションを、Amazon Rekognition .NET API を使用して作成する方法を示します。アプリケーションは、Amazon Simple Email Service (Amazon SES) を使用して、結果を記載した E メール通知を管理者に送信します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK for Java 2.x

Amazon Rekognition Java API を使用して、Amazon Rekognition を使用し、 Amazon Simple Storage Service (Amazon S3) バケットにあるイメージ内でカテゴリ別にオブジェクトを識別するアプリケーションを作成する方法について説明します。アプリケーションは、Amazon Simple Email Service (Amazon SES) を使用して、結果を記載した E メール通知を管理者に送信します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition

- Amazon S3
- Amazon SES

JavaScript

SDK for JavaScript (v3)

AWS SDK for JavaScript を含む Amazon Rekognition を使用して、Amazon Rekognition を使用し、 Amazon Simple Storage Service (Amazon S3) バケット 内にあるイメージ内でカテゴリ別にオブジェクトを識別するアプリケーションを作成します。アプリケーションは、Amazon Simple Email Service (Amazon SES) を使用して、結果を記載した E メール通知を管理者に送信します。

以下ではその方法を説明しています。

- Amazon Cognito を使用して認証されていないユーザーを作成します。
- Amazon Rekognition を使用して、オブジェクトのイメージを分析します。
- Amazon SES の E メールアドレスを検証します。
- Amazon SES を使用して、E メール通知を送信します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK for Kotlin

Amazon Simple Storage Service (Amazon S3) バケットにある画像内からカテゴリ別にオブジェクトを Amazon Rekognition を使用して識別するアプリケーションを、Amazon Rekognition Kotlin API を使用して作成する方法を示します。アプリケーションは、Amazon Simple Email Service (Amazon SES) を使用して、結果を記載した E メール通知を管理者に送信します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK for Python (Boto3)

AWS SDK for Python (Boto3) を使用して、以下のことができるウェブアプリケーションを作成する方法を示します。

- Amazon Simple Storage Service (Amazon S3) バケットに、写真をアップロードします。
- Amazon Rekognition を使用して、写真を分析およびラベル付けします。
- Amazon Simple Email Service (Amazon SES) を使用して、イメージ分析の E メールレポートを送信します。

この例には、React で構築された JavaScript で記述されたウェブページと、Flask-RESTful で構築された Python で記述された REST サービスの 2 つの主要なコンポーネントが含まれています。

React ウェブページを使用すると、次のことができます。

- S3 バケットに保存されているイメージのリストを表示します。
- イメージを S3 バケットにアップロードします。
- イメージ内で検出された項目を識別するイメージとラベルを表示します。
- S3 バケット内のすべてのイメージのレポートを取得し、レポートの E メールを送信します。

ウェブページが REST サービスを呼び出します。サービスはリクエストを AWS に送信して、以下のアクションを実行します。

- S3 バケット内のイメージのリストを取得し、フィルタリングします。
- Amazon S3 バケットに写真をアップロードします。

- Amazon Rekognition を使用して個々の写真を分析し、写真で検出された項目を識別するラベルのリストを取得します。
- S3 バケット内のすべての写真を分析し、Amazon SES を使用してレポートを E メールで送信します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して、Amazon Rekognition で、動画内の人やオブジェクトを検出する

次のコード例は、Amazon Rekognition で動画内の人やオブジェクトを検出する方法を示します。

Java

SDK for Java 2.x

Amazon Rekognition Java API を使用し、Amazon Simple Storage Service (Amazon S3) バケットにあるビデオ内で顔やオブジェクトを検出するアプリケーションを作成する方法について説明します。アプリケーションは、Amazon Simple Email Service (Amazon SES) を使用して、結果を記載した E メール通知を管理者に送信します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3

- Amazon SES

JavaScript

SDK for JavaScript (v3)

AWS SDK for JavaScript を含む Amazon Rekognition を使用し、Amazon Simple Storage Service (Amazon S3) バケットにあるビデオ内で顔やオブジェクトを検出するアプリケーションを作成します。アプリケーションは、Amazon Simple Email Service (Amazon SES) を使用して、結果を記載した E メール通知を管理者に送信します。

以下ではその方法を説明しています。

- Amazon Cognito を使用して認証されていないユーザーを作成します。
- Amazon Rekognition を使用して、PPE の画像を分析します。
- Amazon SES の E メールアドレスを検証します。
- Amazon SES を使用して、E メール通知を送信します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して、EXIF およびその他のイメージ情報を保存します

次のコードサンプルは、以下の操作方法を示しています。

- JPG、JPEG、または PNG ファイルから EXIF 情報を取得します。
- Amazon S3 バケットにイメージファイルをアップロードします。
- Amazon Rekognition を使用して、ファイル内の 3 つの上位属性 (ラベル) を特定します。

- EXIF およびラベル情報を、リージョンの Amazon DynamoDB テーブルに追加します。

Rust

SDK for Rust

JPG、JPEG、または PNG ファイルから EXIF 情報を取得し、イメージファイルを Amazon S3 バケットにアップロードし、Amazon Rekognition を使用してファイル内の 3 つの上位属性 (Amazon Rekognition のラベル) を特定し、リージョンの Amazon DynamoDB テーブルに EXIF およびラベル情報を追加します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- DynamoDB
- Amazon Rekognition
- Amazon S3

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

S3 Object Lambda でアプリケーションのデータを変換する

次のコード例では、S3 Object Lambda でアプリケーションのデータを変換する方法を示しています。

.NET

AWS SDK for .NET

オブジェクトがリクエストしたクライアントまたはアプリケーションのニーズに合うように、標準の S3 GET リクエストにカスタムコードを追加し、S3 から取得したリクエストされたオブジェクトを変更する方法を示しています。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Lambda
- Amazon S3

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「このサービスを AWS SDK で使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

トラブルシューティング

このセクションでは、Amazon S3 機能をトラブルシューティングする方法と、AWS Support に連絡する際に必要なリクエスト ID を取得する方法について説明します。

トピック

Amazon S3 でのアクセス拒否 (403 Forbidden) エラーのトラブルシューティング

> **Important**
>
> 2024 年 5 月 13 日に、バケット所有者によって開始されていない不正なリクエストに対する料金を排除する変更のデプロイを開始しました。この変更のデプロイが完了すると、バケット所有者は、個々の AWS アカウントまたは AWS 組織の外部からリクエストが開始されたときに `AccessDenied` (HTTP `403 Forbidden`) エラーを返すリクエストに対してリクエストまたは帯域幅料金を請求されることはありません。請求されない HTTP `3XX` および `4XX` ステータスコードの全一覧については、「Amazon S3 のエラーレスポンスに対する請求」を参照してください。この請求の変更は、アプリケーションの更新を必要とせず、すべての S3 バケットに適用されます。すべての AWS リージョンでこの変更のデプロイが完了したら、ドキュメントを更新します。

以下のトピックでは、Amazon S3 でのアクセス拒否 (403 Forbidden) エラーの最も一般的な原因について説明します。

> **Note**
>
> `Access Denied` (HTTP `403 Forbidden`) の場合、リクエストがバケット所有者の個々の AWS アカウントまたはバケット所有者の AWS 組織外で開始されても、S3 ではバケット所有者への請求を行いません。

トピック

- バケットポリシーと IAM ポリシー
- Amazon S3 ACL 設定
- S3 ブロックパブリックアクセス設定
- Amazon S3 の暗号化設定
- S3 オブジェクトロック設定
- VPC エンドポイントポリシー
- AWS Organizations ポリシー
- アクセスポイント設定

> **Note**
>
> アクセス許可の問題のトラブルシューティングを行う場合は、「バケットポリシーと IAM ポリシー」セクションから始めて、「アクセス許可を確認するためのヒント」のガイダンスに必ず従ってください。

バケットポリシーと IAM ポリシー

バケットレベルの操作

バケットポリシーが設定されていない場合、バケットはバケット所有アカウントの任意のAWS Identity and Access Management (IAM) ID からのリクエストを暗黙的に許可します。また、バケットは、他の任意のアカウントからの他の IAM ID からのリクエストや、匿名の (署名されていない) リクエストを暗黙的に拒否します。ただし、IAM ユーザーポリシーが設定されていない場合、リクエスタは (ルートユーザーでない限り) リクエストを行うことを暗黙的に拒否されます。詳細については、IAM ユーザーガイドの「アカウント内でのリクエストの許可または拒否の決定」を参照してください。

オブジェクトレベルの操作

バケット所有アカウントがオブジェクトを所有している場合、バケットポリシーと IAM ユーザーポリシーは、オブジェクトレベルの操作でもバケットレベルの操作と同じように機能します。例えば、バケットポリシーが設定されていない場合、バケットはバケット所有アカウントの任意の IAM ID からのオブジェクトリクエストを暗黙的に許可します。また、バケットは、他の任意のアカウントの他の IAM ID からのオブジェクトリクエストや、匿名 (署名なし) リクエストを暗黙的に拒否します。ただし、IAM ユーザーポリシーが設定されていない場合、リクエスタは (ルートユーザーでない限り) オブジェクトリクエストを行うことを暗黙的に拒否されます。

オブジェクトが外部アカウントによって所有されている場合、オブジェクトへのアクセスはオブジェクトアクセスコントロールリスト (ACL) を通じてのみ許可されます。バケットポリシーと IAM ユーザーポリシーは、引き続きオブジェクトリクエストを拒否するために使用できます。

したがって、バケットポリシーまたは IAM ユーザーポリシーがアクセス拒否 (403 Forbidden) エラーを引き起こしていないことを確認するには、次の要件が満たされていることを確認してください。

- 同じアカウントでアクセスする場合、バケットポリシーまたは IAM ユーザーポリシーのいずれにも、アクセス権限を付与するリクエスタに対する明示的な `Deny` 記述があってはなりません。バケットポリシーと IAM ユーザーポリシーのみを使用してアクセス権限を付与する場合は、これらのポリシーの 1 つに少なくとも 1 つの明示的な `Allow` ステートメントが必要です。
- クロスアカウントアクセスの場合、バケットポリシーまたは IAM ユーザーポリシーのいずれにも、アクセス権限を付与するリクエスタに対する明示的な `Deny` 記述があってはなりません。バケットポリシーと IAM ユーザーポリシーのみを使用してクロスアカウントアクセス権限を付与する場合は、リクエスタのバケットポリシーと IAM ユーザーポリシーの両方に明示的な `Allow` ステートメントを含める必要があります。

> **Note**
>
> バケットポリシーの `Allow` ステートメントは、同じバケット所有アカウントが所有するオブジェクトにのみ適用されます。ただし、バケットポリシーの `Deny` ステートメントは、オブジェクトの所有権に関係なくすべてのオブジェクトに適用されます。

バケットポリシーを確認または編集する方法

> **Note**
>
> バケットポリシーを表示または編集するには、`s3:GetBucketPolicy` アクセス許可が必要です。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、バケットポリシーを表示または編集するバケットの名前を選択します。
4. [アクセス許可] タブを選択します。
5. [バケットポリシー] で [編集] を選択します。[バケットポリシーを編集] ページが表示されます。

AWS Command Line Interface (AWS CLI) を使用してバケットポリシーを確認または編集するには、get-bucket-policy コマンドを使用します。

> **Note**
>
> バケットポリシーが正しくないためにバケットからロックアウトされた場合は、ルートユーザーの認証情報を使用して AWS Management Console にサインインします。バケットへのアクセスを回復するには、必ずルートユーザーの認証情報を使用してバケットポリシーを削除してください。

アクセス許可を確認するためのヒント

リクエスタが Amazon S3 オペレーションを実行するための適切な権限を持っているかどうかを確認するには、以下を試してください。

- リクエスタを特定します。署名なしのリクエストの場合、IAM ユーザーポリシーのない匿名リクエストです。署名済み URL を使用したリクエストの場合、ユーザーポリシーは、リクエストに署名した IAM ユーザーまたはロールのポリシーと同じになります。
- 正しい IAM ユーザーまたはロールを使用していることを確認します。IAM ユーザーまたはロールを確認するには、AWS Management Console の右上隅を確認するか、aws sts get-caller-identity コマンドを使用することができます。

- IAM ユーザーまたはロールに関連付けられている IAM ポリシーを確認します。次のいずれかの方法を使用します。
  - IAM ポリシーシミュレーターを使用して IAM ポリシーをテストします。
  - さまざまな IAM ポリシータイプを確認します。
- 必要に応じて、IAM ユーザーポリシーを編集します。
- アクセスを明示的に拒否または許可するポリシーについては、以下の例を確認してください。
  - IAM ユーザーポリシーの明示的な許可: IAM: 複数のサービスへのアクセスをプログラムおよびコンソールで許可および拒否
  - バケットポリシーの明示的許可: オブジェクトをアップロードしたり、パブリックアクセス用のオブジェクト ACL を設定したりする権限を複数のアカウントに付与
  - IAM ユーザーポリシーの明示的拒否: AWS: リクエストされた AWS リージョン に基づいてAWS へのアクセスを拒否
  - バケットポリシーの明示的拒否: バケットに書き込まれるすべてのオブジェクトに SSE-KMS が必要

Amazon S3 ACL 設定

ACL 設定を確認するときは、まずオブジェクト所有権の設定を確認して、バケットで ACL が有効になっているかどうかを確認してください。ACL 権限は権限を付与するためにのみ使用でき、リクエストを拒否するためには使用できないことに注意してください。また、バケットポリシーや IAM ユーザーポリシーでの明示的拒否によって拒否されたリクエスタにアクセス権を付与するために ACL を使用することもできません。

オブジェクト所有権設定にバケット所有者強制を設定

バケット所有者強制設定が有効になっている場合、この設定によってバケットとオブジェクトに適用されるすべての ACL が無効になるため、ACL 設定によってアクセス拒否 (403 Forbidden) エラーが発生する可能性はほとんどありません。Amazon S3 バケットのデフォルト (および推奨) 設定は、バケット所有者強制です。

オブジェクト所有権設定にバケット所有者優先またはオブジェクト作成者を設定

ACL 権限は、バケット所有者優先設定またはオブジェクト作成者設定でも引き続き有効です。ACL には、バケット ACL とオブジェクト ACL の 2 種類があります。この 2 種類の ACL の違いについては、「ACL アクセス許可とアクセスポリシーのアクセス許可のマッピング」を参照してください。

拒否されたリクエストのアクションに応じて、以下のようにバケットまたはオブジェクトの ACL 権限を確認してください。

- Amazon S3 が LIST、PUT オブジェクト、GetBucketAcl、または PutBucketAcl リクエストを拒否した場合は、バケットの ACL 権限を確認してください。

> **Note**
>
> バケット ACL 設定で GET オブジェクト権限を付与することはできません。

- Amazon S3 が S3 オブジェクトの GET リクエスト、または PutObjectAcl リクエストを拒否した場合は、そのオブジェクトの ACL 権限を確認してください。

> **Important**
>
> オブジェクトを所有するアカウントがバケットを所有するアカウントと異なる場合、オブジェクトへのアクセスはバケットポリシーによって制御されません。

クロスアカウントによるオブジェクト所有権である場合の、GET オブジェクトリクエストによるアクセス拒否 (403 Forbidden) エラーのトラブルシューティング

バケットのオブジェクト所有権設定を確認して、オブジェクト所有者を判断します。オブジェクト ACL へのアクセス権限がある場合は、オブジェクト所有者のアカウントを確認することもできます。(オブジェクト所有者のアカウントを表示するには、Amazon S3 コンソールでオブジェクト ACL 設定を確認します)。または、GetObjectAcl リクエストを実行してオブジェクト所有者の正規 ID を検索し、オブジェクト所有者のアカウントを確認することもできます。デフォルトでは、ACL はオブジェクト所有者のアカウントへの GET リクエストに対する明示的な許可権限を付与します。

オブジェクト所有者がバケット所有者と異なることを確認してから、ユースケースとアクセスレベルに応じて、次のいずれかの方法を選択してアクセス拒否 (403 Forbidden) エラーに対処します。

- ACL を無効にする (推奨) - この方法はすべてのオブジェクトに適用され、バケット所有者が実行できます。この方法では、バケット所有者に所有権が自動的に与えられ、バケット内のすべてのオブジェクトを完全に制御できます。この方法を実行する前に、ACL を無効にする前提条件を確認します。バケット所有者強制 (推奨) モードにバケットを設定する方法については、「既存のバケットでのオブジェクトの所有権の設定」を参照してください。

> **Important**
>
> アクセス拒否 (403 Forbidden) エラーを防ぐには、ACL を無効にする前に、必ず ACL 権限をバケットポリシーに移行してください。詳細については、「ACL 権限からの移行に関するバケットポリシーの例」を参照してください。

- オブジェクト所有者をバケット所有者に変更 - この方法は個々のオブジェクトに適用できますが、オブジェクトの所有権を変更できるのはオブジェクト所有者 (または適切な権限を持つユーザー) のみです。追加の PUT コストが適用される場合があります。(詳細については、「Amazon S3 の料金」を参照してください)。この方法では、バケット所有者にオブジェクトの完全な所有権が付与され、バケット所有者はバケットポリシーを通じてオブジェクトへのアクセスを制御できます。

  オブジェクトの所有権を変更するには、以下のいずれかを実行します。

  - バケット所有者はオブジェクトをコピーしてバケットに戻すことができます。
  - バケットのオブジェクト所有権設定を「バケット所有者優先」に変更できます。バージョニングを無効にすると、バケット内のオブジェクトは上書きされます。バージョニングが有効になっている場合、同じオブジェクトの重複バージョンがバケットに表示され、バケット所有者はライフサイクルルールを期限切れに設定できます。オブジェクト所有権設定を変更する方法については、「既存のバケットでのオブジェクトの所有権の設定」を参照してください。

    > **Note**
    >
    > オブジェクト所有権の設定をバケット所有者優先に更新すると、その設定はバケットにアップロードされた新しいオブジェクトにのみ適用されます。

  - `bucket-owner-full-control` 既定オブジェクト ACL を使用して、オブジェクト所有者にオブジェクトを再度アップロードさせることができます。

  > **Note**
  >
  > クロスアカウントアップロードの場合、バケットポリシーで `bucket-owner-full-control` 既定オブジェクト ACL を要求することもできます。バケットポリシーの例については、バケット所有者はフルコントロール権限を持ちながら、オブジェクトをアップロードするためのクロスアカウントアクセス許可を付与するを参照してください。

- オブジェクト作成者をオブジェクト所有者のままにする - この方法ではオブジェクトの所有者は変更されませんが、オブジェクトへのアクセス権限を個別に付与できます。オブジェクトへのアクセ

ス権限を付与するには、そのオブジェクトに対する PutObjectAcl 権限が必要です。次に、アクセス拒否 (403 Forbidden) エラーを修正するために、リクエスタをオブジェクトの ACL 内のオブジェクトへのアクセス権限付与者として追加します。詳細については、「ACL の設定」を参照してください。

S3 ブロックパブリックアクセス設定

失敗したリクエストにパブリックアクセスまたはパブリックポリシーが含まれる場合は、アカウント、バケット、または S3 アクセスポイントの S3 ブロックパブリックアクセス設定を確認してください。2023 年 4 月以降、デフォルトで新しいバケットに対してすべてのブロックパブリックアクセス設定が有効になります。Amazon S3 が「パブリック」を定義する方法の詳細については、「「パブリック」の意味」を参照してください。

TRUE に設定すると、ブロックパブリックアクセス設定は明示的な拒否ポリシーとして機能し、ACL、バケットポリシー、IAM ユーザーポリシーで許可されている権限よりも優先されます。ブロックパブリックアクセス設定がリクエストを拒否しているかどうかを判断するには、以下のシナリオを確認してください。

- 指定されたアクセスコントロールリスト (ACL) がパブリックである場合は、BlockPublicAcls 設定によって PutBucketAcl および PutObjectACL コールが拒否されます。
- リクエストにパブリック ACL が含まれている場合は、BlockPublicAcls 設定によって PutObject コールが拒否されます。
- あるアカウントに BlockPublicAcls 設定が適用され、リクエストにパブリック ACL が含まれている場合は、パブリック ACL を含む CreateBucket コールはすべて失敗します。
- リクエストの許可がパブリック ACL によってのみ付与されている場合は、IgnorePublicAcls 設定によってリクエストが拒否されます。
- 指定されたバケットポリシーでパブリックアクセスが許可されている場合は、BlockPublicPolicy 設定によって PutBucketPolicy コールが拒否されます。
- BlockPublicPolicy 設定がアクセスポイントに適用されている場合は、パブリックポリシーを指定し、アクセスポイント経由で実行される PutAccessPointPolicy および PutBucketPolicy コールは失敗します。
- アクセスポイントまたはバケットにパブリックポリシーが設定されている場合は、RestrictPublicBuckets 設定によって AWS のサービス プリンシパルを除くすべてのクロスアカウントコールが拒否されます。この設定では、匿名 (または署名なし) のコールもすべて拒否されます。

ブロックパブリックアクセスの設定を確認および更新するには、「S3 バケットへのパブリックアクセスブロック設定の構成」を参照してください。

Amazon S3 の暗号化設定

Amazon S3 はバケットのサーバー側暗号化をサポートしています。サーバー側の暗号化とは、データを受信するアプリケーションまたはサービスによって、送信先でデータを暗号化することです。Amazon S3 は、AWS データセンターのディスクに書き込まれるときにデータをオブジェクトレベルで暗号化し、お客様がデータにアクセスするときに復号します。

デフォルトで、Amazon S3 では、Amazon S3 のすべてのバケットに対する基本レベルの暗号化として、Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) が適用されるようになりました。Amazon S3 では、オブジェクトをアップロードするときにサーバー側を暗号化する方法も指定できます。

バケットのサーバー側暗号化ステータスと暗号化設定を確認する方法

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、暗号化設定を確認するバケットを選択します。
4. プロパティ タブを選択します。
5. [デフォルトの暗号化] セクションまでスクロールして、[暗号化タイプ] 設定を表示します。

AWS CLI を使用して暗号化設定を確認するには、get-bucket-encryption コマンドを使用します。

オブジェクトの暗号化状態を確認する方法

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、オブジェクトが含まれるバケットの名前を選択します。
4. [オブジェクト] リストで、暗号化を追加または変更するオブジェクトの名前を選択します。

   オブジェクトの詳細ページが表示されます。
5. [サーバー側の暗号化設定] セクションまでスクロールすると、オブジェクトのサーバー側暗号化設定が表示されます。

AWS CLI を使用してオブジェクト暗号化ステータスを確認するには、head-object コマンドを使用します。

暗号化とアクセス許可の要件

Amazon S3 では 3 種類のサーバー側暗号化がサポートされます。

- Amazon S3 マネージドキーを用いたサーバー側の暗号化 (SSE-S3)
- AWS Key Management Service (AWS KMS) キー (SSE-KMS) によるサーバー側の暗号化
- 顧客提供のキーを用いたサーバー側の暗号化 (SSE-C)。

暗号化設定に基づいて、以下のアクセス許可要件が満たされていることを確認します。

- SSE-S3 - 追加の権限は必要ありません。
- SSE-KMS (カスタマー管理キーを使用) - オブジェクトをアップロードするには、AWS KMS key に `kms:GenerateDataKey` 権限が必要です。オブジェクトをダウンロードしてオブジェクトのマルチパートアップロードを実行するには、KMS キーに `kms:Decrypt` 権限が必要です。
- SSE-KMS (AWS マネージドキー を使用) - リクエスタは `aws/s3` KMS キーを所有しているのと同じアカウントに属している必要があります。また、リクエスタはオブジェクトにアクセスするための正しい Amazon S3 権限を持っている必要があります。
- SSE-C (お客様提供のキーを使用) - 追加の権限は必要ありません。バケットポリシーを設定すると、バケットのオブジェクトに対してお客様が用意した暗号化キーでサーバー側の暗号化をリクエストおよび制限できます。

オブジェクトがカスタマーマネージドキーで暗号化されている場合は、KMS キーポリシーによって `kms:GenerateDataKey` または `kms:Decrypt` アクションの実行が許可されていることを確認してください。KMS キーポリシーを確認する手順については、AWS Key Management Service デベロッパーガイドの「キーポリシーの表示」を参照してください。

S3 オブジェクトロック設定

バケットで S3 オブジェクトロックが有効で、オブジェクトが保持期間またはリーガルホールドで保護されている場合、オブジェクトを削除しようとすると Amazon S3 はアクセス拒否 (403 Forbidden) エラーを返します。

バケットでオブジェクトロックが有効になっているかどうかを確認する方法

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、確認するバケットの名前を選択します。
4. プロパティ タブを選択します。
5. [オブジェクトロック] セクションまでスクロールします。[オブジェクトロック] 設定が [有効] または [無効] であるかを確認します。

オブジェクトが保持期間またはリーガルホールドで保護されているかを判断するには、オブジェクトのロック情報を参照します。

オブジェクトが保持期間またはリーガルホールドで保護されている場合は、次の点を確認します。

- オブジェクトバージョンがコンプライアンス保存モードで保護されている場合、それを完全に削除する方法はありません。ルートユーザーを含む、任意のリクエスタからの永続的な DELETE リクエストは、アクセス拒否 (403 Forbidden) エラーになります。また、コンプライアンス保持モードで保護されているオブジェクトの DELETE リクエストを送信すると、Amazon S3 はそのオブジェクトの削除マーカーを作成することに注意してください。
- オブジェクトバージョンがガバナンス保持モードで保護されていて `s3:BypassGovernanceRetention` 権限がある場合は、保護をバイパスしてバージョンを完全に削除できます。詳細については、「ガバナンスモードのバイパス」を参照してください。
- オブジェクトバージョンがリーガルホールドで保護されている場合は、永久 DELETE リクエストを実行するとアクセス拒否 (403 Forbidden) エラーが発生する可能性があります。オブジェクトバージョンを永久に削除するには、オブジェクトバージョンのリーガルホールドを解除する必要があります。リーガルホールドを解除するには、`s3:PutObjectLegalHold` 許可が必要です。リーガルホールドの解除の詳細については、「オブジェクトロックの設定」を参照してください。

VPC エンドポイントポリシー

仮想プライベートクラウド (VPC) エンドポイントを使用して Amazon S3 にアクセスする場合は、VPC エンドポイントポリシーによって Amazon S3 リソースへのアクセスがブロックされていないことを確認します。デフォルトでは、VPC エンドポイントポリシーは Amazon S3 へのすべてのリクエストを許可します。特定のリクエストを制限するように VPC エンドポイントポリシーを設

定することもできます。VPC エンドポイントポリシーを確認する方法については、AWS PrivateLink ガイドの「エンドポイントポリシーを使用して VPC エンドポイントへのアクセスを制御する」を参照してください。

AWS Organizations ポリシー

あなたの AWS アカウント が組織に所属している場合、AWS Organizations ポリシーによって Amazon S3 リソースへのアクセスがブロックされる場合があります。デフォルトでは、AWS Organizations ポリシーは Amazon S3 へのリクエストをブロックしません。ただし、AWS Organizations ポリシーが S3 バケットへのアクセスをブロックするように設定されていないことを確認してください。AWS Organizations ポリシーを確認する手順については、AWS Organizations ユーザーガイドの「すべてのポリシーの一覧表示」を参照してください。

アクセスポイント設定

Amazon S3 Access Points 経由でリクエストを行っているときにアクセス拒否 (403 Forbidden) エラーが表示される場合は、次の点を確認する必要がある場合があります。

- アクセスポイントの設定
- アクセスポイントに使用される IAM ユーザーポリシー
- クロスアカウントアクセスポイントの管理または設定に使用されるバケットポリシー

アクセスポイントの設定とポリシー

- アクセスポイントを作成するときに、インターネットまたは VPC をネットワークオリジンとして指定できます。ネットワークオリジンが VPC のみに設定されている場合、Amazon S3 は指定 VPC 以外のアクセスポイントへのリクエストをすべて拒否します。アクセスポイントのネットワークオリジンを確認するには、Virtual Private Cloud に制限されたアクセスポイントの作成 を参照してください。
- アクセスポイントでは、カスタムのブロックパブリックアクセス設定を構成することもできます。これは、バケットレベルまたはアカウントレベルのブロックパブリックアクセス設定と同様に機能します。カスタムのブロックパブリックアクセス設定を確認する方法は、「アクセスポイントへのパブリックアクセスの管理」を参照してください。
- アクセスポイントを使用して Amazon S3 へのリクエストを正常に行うには、リクエスタが必要な IAM 権限を持っていることを確認します。詳細については、「アクセスポイントを使用するための IAM ポリシーの設定」を参照してください。

- リクエストにクロスアカウントアクセスポイントが含まれる場合、バケット所有者がアクセスポイントからのリクエストを許可するようにバケットポリシーを更新済みであることを確認します。詳細については、「クロスアカウントアクセスポイントへのアクセス許可の付与」を参照してください。

このトピックのすべての項目を確認してもアクセス拒否 (403 Forbidden) エラーが解決しない場合は、Amazon S3 リクエスト ID を取得して、追加ガイダンスについて AWS Support まで問い合わせてください。

バッチオペレーションのトラブルシューティング

次のトピックは、バッチオペレーション中に発生する可能性がある問題のトラブルシューティングに役立ちます。

共通エラー

- アクセス許可の問題があるか、S3 オブジェクトロック保持モードが有効の場合に、ジョブレポートが配信されない
- S3 バッチレプリケーションがエラーで失敗しました。マニフェストの生成で、フィルター条件に一致するキーが見つかりませんでした。
- バッチオペレーションの失敗は、既存のレプリケーション設定に新しいレプリケーションルールを追加した後に発生します
- 「400 InvalidRequest: VersionId がないためにタスクが失敗しました」によりオブジェクトが失敗したバッチオペレーション
- ジョブのタグオプションが有効な状態でジョブの失敗を作成する
- マニフェストの読み取り拒否

アクセス許可の問題があるか、S3 オブジェクトロック保持モードが有効の場合に、ジョブレポートが配信されない

次のエラーは、必要なアクセス許可がないか、配信先のバケットでオブジェクトロックの保持モード (ガバナンスモードまたはコンプライアンスモード) が有効になっている場合に発生します。

エラー: 失敗の理由。ジョブレポートをレポートバケットに書き込めませんでした。権限を確認してください。

IAM ロールと信頼ポリシーは、レポートの配信先であるバケットの PUT オブジェクトに対するアクセスを S3 バッチオペレーションに許可するように設定する必要があります。必要なアクセス許可がないと、ジョブレポートの配信エラーが発生します。

保持モードが有効な場合、バケットは Write-Once-Read-Many (WORM) で保護されます。オブジェクトロックの保持モードが有効になっている配信先のバケットはサポートされないため、ジョブ完了レポートの配信試行は失敗します。この問題を解決するには、オブジェクトロック保持モードが有効になっていないジョブ完了レポートのレプリケート先バケットを選択します。

S3 バッチレプリケーションがエラーで失敗しました。マニフェストの生成で、フィルター条件に一致するキーが見つかりませんでした。

エラー: マニフェストの生成で、フィルター条件に一致するキーが見つかりませんでした。

このエラーは、以下のいずれかの原因で発生することがあります。

- レプリケート元バケット内のオブジェクトが S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive のストレージクラスに保存されている場合。

  これらのオブジェクトでバッチレプリケーションを使用するには、まず、バッチオペレーションジョブで S3 オブジェクトの復元開始オペレーションを使用して、オブジェクトを S3 標準ストレージクラスに復元します。詳細については、「アーカイブされたオブジェクトの復元」および「オブジェクトの復元 (バッチオペレーション)」を参照してください。オブジェクトの復元後、バッチレプリケーションジョブを使用してオブジェクトをレプリケートできます。
- 指定したフィルター条件がソースバケット内の有効なオブジェクトと一致しない場合。

  フィルター条件を確認して修正します。例えば、バッチレプリケーションルールでは、フィルター条件は *example-s3-bucket* 内のすべてのオブジェクトをプレフィックス Tax/ で検索しています。プレフィックス名が誤って入力され、スラッシュが末尾だけでなく、/Tax/ の先頭と末尾の両方にある場合、S3 オブジェクトは見つかりません。このエラーを解決するには、レプリケーションルールのプレフィックスを /Tax/ から Tax/ に修正します。

バッチオペレーションの失敗は、既存のレプリケーション設定に新しいレプリケーションルールを追加した後に発生します

バッチオペレーションは、レプリケート元バケットのレプリケーション設定のすべてのルールに対して、既存オブジェクトのレプリケーションを実行しようとします。既存のレプリケーションルールのいずれかに問題があると、失敗が発生する可能性があります。

バッチオペレーションジョブの完了レポートには、ジョブの失敗理由が説明されています。一般的なエラーのリストについては、「Amazon S3 レプリケーションの失敗の理由」を参照してください。

「400 InvalidRequest: VersionId がないためにタスクが失敗しました」によりオブジェクトが失敗したバッチオペレーション

バッチオペレーションジョブがバージョン管理されたバケット内のオブジェクトに対してアクションを実行している場合、マニフェスト内でバージョン ID フィールドが空のオブジェクトがあると、次の例のようなエラーが発生します。

エラー: *BUCKET_NAME,prefix/file_name*,failed,400,InvalidRequest,VersionId がないためにタスクが失敗しました

このエラーは、マニフェストのバージョン ID フィールドが null リテラル文字列ではなく、空の文字列であるために発生します。

バッチオペレーションは、その特定のオブジェクトでは失敗しますが、ジョブ全体では失敗しません。この問題は、マニフェスト形式がオペレーション中にバージョン ID を使用するように設定されている場合に発生します。バージョン管理されていないジョブでは、各オブジェクトの最新バージョンでのみ動作し、マニフェストのバージョン ID は無視されるため、この問題は発生しません。

この問題を解決するには、空のバージョン ID を null 文字列に変換します。詳細については、「the section called “空のバージョン ID 文字列を NULL 文字列に変換します。”」を参照してください。

ジョブのタグオプションが有効な状態でジョブの失敗を作成する

`s3:PutJobTagging` 権限がない場合、ジョブのタグオプションを有効にしてバッチオペレーションジョブを作成すると `403 access denied` エラーが発生します。

ジョブのタグオプションを有効にしてバッチオペレーションジョブを作成するには、バッチオペレーションジョブを作成する AWS Identity and Access Management (IAM) ユーザーに、`s3:PutJobTagging` 権限に加えて `s3:CreateJob` 権限が必要です。

バッチオペレーションに必要なアクセス許可の詳細については、「the section called “アクセス許可の付与”」を参照してください。

マニフェストの読み取り拒否

バッチオペレーションジョブを作成しようとしたときに、バッチオペレーションがマニフェストファイルを読み取れない場合、次のエラーが発生する可能性があります。

AWS CLI

失敗の理由: マニフェストの読み取りは禁止されています: AccessDenied

Amazon S3 コンソール

警告: マニフェストオブジェクトの ETag を取得できません。別のオブジェクトを指定して続行してください。

この問題を解決するには、次の手順を実行します。

- バッチオペレーションジョブの作成に使用した AWS アカウント の IAM ロールに `s3:GetObject` 権限があることを確認します。アカウントの IAM ロールには、バッチオペレーションがマニフェストファイルの読み取りを許可する `s3:GetObject` 権限が必要です。

  バッチオペレーションに必要なアクセス許可の詳細については、「the section called “アクセス許可の付与”」を参照してください。
- マニフェストオブジェクトのメタデータに、S3 オブジェクトの所有権とのアクセスの不一致がないか確認します。S3 オブジェクトの所有権の詳細については、「the section called “オブジェクト所有者の管理”」を参照してください。
- マニフェストファイルの暗号化に AWS Key Management Service (AWS KMS) キーが使用されているかどうかを確認してください。

  バッチオペレーションでは、AWS KMS で暗号化された CSV 形式のインベントリレポートがサポートされています。ただし、バッチオペレーションでは、AWS KMS で暗号化された CSV 形式のマニフェストファイルはサポートされていません。詳細については、Amazon S3 インベントリの設定およびマニフェストの指定を参照してください。

CORS のトラブルシューティング

CORS 設定を行ったバケットにアクセス中に予期しない動作が発生した場合は、以下のステップを試してトラブルシューティングを行います。

1. バケットに CORS 設定が行われていることを確認します。

   CORS 設定を行うと、コンソールでは、[Properties] (プロパティ) バケットの [Permissions] (アクセス権限) セクションに [Edit CORS Configuration] (CORS 設定の編集) リンクが表示されます。

2. 任意のツールを使用してリクエストとレスポンスの全体を取り込みます。Amazon S3 が受信するすべてのリクエストには、リクエストのデータと一致する、以下のような CORS ルールが存在している必要があります。

   a. リクエストに Origin ヘッダーがあること。

      ヘッダーがない場合は、Amazon S3 でクロスオリジンリクエストとして扱われず、CORS レスポンスヘッダーをレスポンスに入れて送信されません。

   b. リクエストの Origin ヘッダーが指定された `AllowedOrigin` の `CORSRule` エレメントの少なくとも 1 つと一致していること。

      リクエストの Origin ヘッダーのスキーム、ホスト、およびポートの値が `AllowedOrigin` の `CORSRule` エレメントと一致していること。例えば、オリジン `CORSRule` を許可するように `http://www.example.com` を設定した場合は、リクエスト内に `https://www.example.com` や `http://www.example.com:80` のオリジンがあると、設定で許可されているオリジンと一致しなくなります。

   c. リクエスト内のメソッド (または、プリフライトリクエストの場合は `Access-Control-Request-Method` に指定したメソッド) が、同じ `AllowedMethod` 内の `CORSRule` エレメントの 1 つであること。

   d. プリフライトリクエストの場合、それに `Access-Control-Request-Headers` ヘッダーが含まれているときに、`CORSRule` に `AllowedHeader` ヘッダーのすべての値に対する `Access-Control-Request-Headers` エントリが含まれていること。

Amazon S3 ライフサイクル問題のトラブルシューティング

以下の情報は、Amazon S3 ライフサイクル問題のトラブルシューティングに役立ちます。

トピック

- バケットでリストオペレーションを実行したところ、有効期限が切れているか、ライフサイクルルールによって移行されたと思われるオブジェクトが表示されました。
- ライフサイクルルールによって実行されたアクションをモニタリングするにはどうすればよいですか？
- バージョニング対応のバケットにライフサイクルルールを設定した後でも、S3 オブジェクトの数は増え続けています。
- ライフサイクルルールを使用して S3 バケットを空にするにはどうすればよいですか?

- オブジェクトを低コストのストレージクラスに移行した後、Amazon S3 の請求額が増加しました。
- バケットポリシーを更新しましたが、S3 オブジェクトが期限切れのライフサイクルルールによってまだ削除されたままです。
- S3 ライフサイクルルールによって期限切れになった S3 オブジェクトを回復できますか?

バケットでリストオペレーションを実行したところ、有効期限が切れているか、ライフサイクルルールによって移行されたと思われるオブジェクトが表示されました。

S3 ライフサイクルオブジェクトの移行とオブジェクトの有効期限は非同期リストオペレーションです。そのため、オブジェクトが期限切れまたは移行の対象となる時点と、実際に移行または期限切れになるまでに遅延が生じる可能性があります。請求の変更は、アクションが完了していなくても、ライフサイクル ルールが満たされるとすぐに適用されます。この動作の例外は、ライフサイクルルールが S3 Intelligent-Tiering ストレージクラスに移行するように設定されている場合です。オブジェクトが S3 Intelligent-Tiering に移行するまで、請求の変更は行われません。請求の変更の詳細については、「バケットのライフサイクル設定を設定する」を参照してください。

> **Note**
>
> Amazon S3 は、128 KB 未満のオブジェクトを S3 標準または S3 標準 - IA ストレージクラスから S3 Intelligent-Tiering、S3 標準 - IA、または S3 1 ゾーン - IA ストレージクラスに移行しません。

ライフサイクルルールによって実行されたアクションをモニタリングするにはどうすればよいですか ?

ライフサイクルルールによって実行されたアクションをモニタリングするには、以下の機能を使用します。

- S3 イベント通知 - S3 ライフサイクルの有効期限や移行イベントが通知されるように S3 イベント通知を設定できます。
- S3 サーバーアクセスログ - S3 バケットのサーバーアクセスログを有効にして、別のストレージクラスへのオブジェクトの移行やオブジェクトの失効など、S3 ライフサイクル関連のアクションをキャプチャできます。詳細については、「ライフサイクルおよびログ記録」を参照してください。

ライフサイクルアクションによるストレージの変更を毎日確認するには、Amazon CloudWatch メトリクスを使用する代わりに S3 Storage Lens ダッシュボードを使用することをお勧めします。Storage Lens ダッシュボードでは、オブジェクト数またはサイズをモニタリングする以下のメトリクスを表示できます。

- 現行バージョンのバイト数
- 現行バージョンのオブジェクト数
- 旧バージョンのバイト数
- 旧バージョンのオブジェクト数
- 削除マーカーのオブジェクト数
- 削除マーカーのストレージバイト数
- 不完全なマルチパートアップロードのバイト数
- 不完全なマルチパートアップロードのオブジェクト数

バージョニング対応のバケットにライフサイクルルールを設定した後でも、S3 オブジェクトの数は増え続けています。

バージョニングが有効なバケットでは、オブジェクトの有効期限が切れた場合でも、オブジェクトはバケットから完全には削除されません。代わりに、オブジェクトの最新バージョンとして削除マーカーが作成されます。削除マーカーは引き続きオブジェクトとしてカウントされます。したがって、現在のバージョンのみを期限切れにするライフサイクルルールを作成した場合、S3 バケット内のオブジェクト数は減少するのではなく実際に増加します。

例えば、S3 バケットで 100 個のオブジェクトでバージョニングが有効になっていて、そのオブジェクトの現在のバージョンが 7 日後に期限切れになるようにライフサイクルルールが設定されているとします。7 日目以降は、元のオブジェクト 100 個 に加え、非現行バージョンとなる削除マーカーが 100 個作成されるため、オブジェクト数は 200 個に増加します。バージョニングが有効なバケットの S3 ライフサイクル設定ルールアクションの詳細については、「バケットのライフサイクル設定を設定する」を参照してください。

オブジェクトを完全に削除するには、ライフサイクル設定を追加して、以前のバージョンのオブジェクト、期限切れの削除マーカー、および不完全なマルチパートアップロードを削除します。新しいライフサイクルルールを作成する方法については、「バケットにライフサイクル設定を設定する」を参照してください。

Note

- Amazon S3 は、オブジェクトの移行日または有効期限を翌日の午前 0 時 (UTC) に丸めます。

  ライフサイクルアクションのオブジェクトを評価する場合、Amazon S3 は UTC でのオブジェクトの作成時間を使用します。例えば、バージョン管理されていないバケットで、1 日後にオブジェクトを期限切れにするように設定されたライフサイクルルールがあるとします。オブジェクトが 1 月 1 日 17:05 太平洋夏時間 (PDT) に作成されたとします。これは UTC では 1 月 2 日 00:05 になります。オブジェクトは 1 月 3 日の 00:05 UTC に 1 日経過し、S3 ライフサイクルが 1 月 4 日の 00:00 UTC にオブジェクトを評価すると、オブジェクトは有効期限の対象となります。

  Amazon S3 ライフサイクルアクションは非同期で実行されるため、ライフサイクルルールで指定された日付とオブジェクトの実際の物理的な移行の間に多少の遅延が生じる場合があります。詳細については、「移行または有効期限の遅延」を参照してください。

  詳細については、「ライフサイクルルール: オブジェクトの存在時間に基づく」を参照してください。
- オブジェクトロックで保護されている S3 オブジェクトの場合、現在のバージョンは完全には削除されません。代わりに、削除マーカーがオブジェクトに追加され、オブジェクトは最新ではなくなります。その後、最新でないバージョンは保存され、永久に期限切れになることはありません。

ライフサイクルルールを使用して S3 バケットを空にするにはどうすればよいですか?

S3 ライフサイクルルールは、何百万ものオブジェクトがある S3 バケットを空にする効果的なツールです。S3 バケットから多数のオブジェクトを削除するには、必ず次の 2 組のライフサイクルルールを使用してください。

- 現在のバージョンのオブジェクトを期限切れにし、以前のバージョンのオブジェクトを完全に削除する
- 期限切れの削除マーカーを削除し、未完了のマルチパートアップロードを削除する

ライフサイクル設定ルールを作成する手順については、「バケットにライフサイクル設定を設定する」を参照してください。

> **Note**
>
> オブジェクトロックで保護されている S3 オブジェクトの場合、現在のバージョンは完全には削除されません。代わりに、削除マーカーがオブジェクトに追加され、オブジェクトは最新ではなくなります。その後、最新でないバージョンは保存され、永久に期限切れになることはありません。

オブジェクトを低コストのストレージクラスに移行した後、Amazon S3 の請求額が増加しました。

オブジェクトを低コストのストレージクラスに移行した後に請求額が増加する理由はいくつかあります。

- 小さなオブジェクトの S3 Glacier オーバーヘッド料金

  S3 Glacier Flexible Retrieval または S3 Glacier Deep Archive に移行した各オブジェクトについて、このストレージ更新に関連する合計 40 KB のオーバーヘッドが発生します。40 KB のオーバーヘッドの一部として、8 KB がメタデータとオブジェクト名の保存に使用されます。この 8 KB は S3 標準料金に従って請求されます。残りの 32 KB はインデックス作成と関連メタデータに使用されます。この 32 KB は、Glacier Flexible Retrieval または S3 Glacier Deep Archive の料金に従って請求されます。

  そのため、小さいサイズのオブジェクトを多数保存する場合は、ライフサイクル移行の使用はお勧めしません。オーバーヘッド料金を削減するには、Amazon S3 に保存する前に、多数の小さなオブジェクトを少数の大きなオブジェクトに集約することを検討してください。コストに関する考慮事項の詳細については、「S3 Glacier Flexible Retrieval と S3 Glacier Deep Archive ストレージクラスへの移行 (オブジェクトのアーカイブ)」を参照してください。

- 最小ストレージ料金

  一部の S3 ストレージクラスには、最小ストレージ期間要件があります。最小期間が満たされる前にそれらのクラスから削除、上書き、または移行されたオブジェクトには、日割り計算による早期移行料金または削除料金が請求されます。これらの最小ストレージ期間要件は次のとおりです。

  - S3 標準 - IA および S3 1 ゾーン - IA - IA - 30 日
  - S3 Glacier Flexible Retrieval および S3 Glacier Instant Retrieval – 90 日

- S3 Glacier Deep Archive – 180 日

これらの要件の詳細については、S3 ライフサイクルを使用したオブジェクトの移行の「制約」セクションを参照してください。一般的な S3 料金情報については、「Amazon S3 の料金表」と「AWS料金計算ツール」を参照してください。

- ライフサイクル移行料金

オブジェクトがライフサイクルルールによって別のストレージクラスに移行されるたびに、Amazon S3 はその移行を 1 つの移行リクエストとしてカウントします。これらの移行リクエストの料金は、これらのストレージクラスの料金に追加されます。多数のオブジェクトを移行する場合は、下位の階層に移行するときに、リクエストの料金を考慮してください。詳細については、「Amazon S3 の料金」を参照してください。

バケットポリシーを更新しましたが、S3 オブジェクトが期限切れのライフサイクルルールによってまだ削除されたままです。

バケットポリシーの Deny ステートメントは、ライフサイクルルールで定義されているオブジェクトの有効期限を妨げません。ライフサイクルアクション (移行や有効期限など) は S3 DeleteObject オペレーションを使用しません。代わりに、S3 ライフサイクルアクションは内部の S3 エンドポイントを使用して実行されます。(詳細については、「ライフサイクルおよびログ記録」を参照してください)

ライフサイクルルールでアクションが実行されないようにするには、ルールを編集、削除、または無効にする必要があります。

S3 ライフサイクルルールによって期限切れになった S3 オブジェクトを回復できますか?

S3 ライフサイクルによって期限切れになったオブジェクトを回復する唯一の方法は、バージョニングを使用することです。バージョニングは、オブジェクトが期限切れの対象となる前に実施する必要があります。ライフサイクルルールによって実行された有効期限オペレーションは元に戻せません。適用されている S3 ライフサイクルルールによってオブジェクトが完全に削除された場合、それらのオブジェクトを復元することはできません。バケットのバージョニングを有効にするには、「the section called “S3 バージョニングの使用”」を参照してください。

バケットにバージョニングを適用しても、オブジェクトの最新バージョンがそのまま残っている場合は、期限切れのオブジェクトの以前のバージョンを復元できます。S3 ライフサイクルルールアク

ションの動作とバージョニングステータスの詳細については、「ライフサイクルアクションを記述する要素」の「ライフサイクルのアクションとバケットのバージョニング状態」テーブルを参照してください。

> **Note**
>
> S3 バケットが AWS バックアップ または S3 レプリケーションによって保護されている場合、これらの機能を使用して期限切れのオブジェクトを回復できる場合もあります。

レプリケーションのトラブルシューティング

このセクションでは、Amazon S3 レプリケーションのトラブルシューティングのヒントと S3 バッチレプリケーションエラーに関する情報を一覧で示します。

トピック

- S3 レプリケーションのトラブルシューティングのヒント
- バッチレプリケーションエラー

S3 レプリケーションのトラブルシューティングのヒント

レプリケーションを設定した後にオブジェクトレプリカがレプリケート先バケットに表示されない場合は、これらのトラブルシューティングのヒントを使用して問題を特定し、修正してください。

- 大部分のオブジェクトは、15 分以内にレプリケートされます。Amazon S3 がオブジェクトをレプリケートするために要する時間は、さまざまな要因に依存します (レプリケート元とレプリケート先のリージョンペア、オブジェクトのサイズなど)。大きなオブジェクトの場合は、レプリケーションには最大で数時間かかることもあります。レプリケート時間を可視化するには、S3 Replication Time Control (S3 RTC) を使用できます。

  レプリケートされているオブジェクトが大きい場合は、レプリケート先バケットに表示されるかどうかを確認する前に、少しの間待機してください。また、レプリケート先オブジェクトのレプリケーションの状態を確認できます。オブジェクトのレプリケーションステータスがPENDINGの場合は、Amazon S3 がレプリケーションを完了していません。オブジェクトのレプリケーションの状態が FAILED の場合は、レプリケート元バケットのレプリケーション設定を確認してください。さらに、レプリケーション中の失敗についての情報を受信するには、Amazon S3 イベント通

知レプリケーションを設定することで失敗イベントを受信できます。詳細については、「Amazon S3 イベント通知によるレプリケーション失敗イベントの受信」を参照してください。

- HeadObject API オペレーションを呼び出して、オブジェクトのレプリケーションの状態を確認できます。HeadObject API オペレーションは、PENDING、COMPLETED、または FAILED で、オブジェクトのレプリケーションの状態を返します。HeadObject API 呼び出しへの応答では、レプリケーションの状態が x-amz-replication-status 要素で返されます。

> **Note**
>
> HeadObject を実行するには、リクエストするオブジェクトへの読み取りアクセス権が必要です。HEAD リクエストには、GET リクエストと同じオプションがあり、GET オペレーションは実行されません。例えば、AWS Command Line Interface (AWS CLI) を使用して HeadObject リクエストを実行するには、次のコマンドを実行します。*user input placeholders* を、ユーザー自身の情報に置き換えます。
>
> ```
> aws s3api head-object --bucket my-bucket --key index.html
> ```

- HeadObject が FAILED のレプリケーション状態のオブジェクトを返したら、S3 バッチレプリケーションを使用して、失敗したオブジェクトをレプリケートできます。また、失敗したオブジェクトをレプリケート元バケットに再アップロードできます。これにより、新しいオブジェクトに対するレプリケーションが開始されます。
- レプリケート元バケットのレプリケーション設定について、以下を確認します。
  - レプリケート先バケットの Amazon リソースネーム (ARN) が正しい。
  - キー名のプレフィックスが正しい。たとえば、Tax というプレフィックスが付いたオブジェクトをレプリケートする設定にした場合、Tax/document1 や Tax/document2 のようなキー名のオブジェクトのみがレプリケートされます。キー名が "document3" のオブジェクトはレプリケートされません。
  - レプリケーションルールの状態は Enabled です。
- レプリケーション設定で、どのバケットでもバージョニングが一時停止されていないことを確認します。レプリケート元とレプリケート先の両方のバケットで、バージョニングを有効にする必要があります。
- レプリケーションルールが [オブジェクト所有権をレプリケート先バケット所有者に変更] に設定されている場合、レプリケーションに使用する AWS Identity and Access Management (IAM) ロールに s3:ObjectOwnerOverrideToBucketOwner 権限が必要です。この権限はリソース (この

場合はレプリケート先バケット) に付与されます。例えば、次の Resource ステートメントは、レプリケート先バケットにこの権限を付与する方法を示しています。

```
{
  "Effect":"Allow",
  "Action":[
    "s3:ObjectOwnerOverrideToBucketOwner"
  ],
  "Resource":"arn:aws:s3:::DestinationBucket/*"
}
```

- レプリケート先バケットを他のアカウントが所有している場合、レプリケート先バケットの所有者は、レプリケート先バケットポリシーにより、レプリケート元バケットの所有者に対しても s3:ObjectOwnerOverrideToBucketOwner 権限を付与する必要があります。次の例のバケットポリシーを使用するには、*user input placeholders* をユーザー自身の情報に置き換えます。

```
{
  "Version": "2012-10-17",
  "Id": "Policy1644945280205",
  "Statement": [
    {
      "Sid": "Stmt1644945277847",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789101:role/s3-replication-role"
      },
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateTags",
        "s3:ObjectOwnerOverrideToBucketOwner"
      ],
      "Resource": "arn:aws:s3:::DestinationBucket/*"
    }
  ]
}
```

> **Note**
>
> レプリケート先バケットのオブジェクト所有権設定に [バケット所有者の強制] が含まれている場合は、レプリケーションルールの [オブジェクト所有権をレプリケート先バケット

所有者に変更] 設定を更新する必要はありません。オブジェクトの所有権の変更はデフォルトで行われます。レプリカの所有権の変更の詳細については、「レプリカ所有者の変更」を参照してください。

- レプリケート元バケットとレプリケート先バケットが異なる AWS アカウント によって所有されているクロスアカウントシナリオでレプリケーション設定を行っている場合、レプリケート先バケットをリクエスタ支払いバケットに設定することはできません。詳細については、「ストレージ転送と使用量のリクエスタ支払いバケットの使用」を参照してください。
- バケットのレプリケート先オブジェクトが AWS Key Management Service (AWS KMS) キーで暗号化されている場合、AWS KMS で暗号化されたオブジェクトが含まれるようにレプリケーションルールを設定する必要があります。Amazon S3 コンソールの暗号化設定で、必ず [AWS KMS で暗号化されたオブジェクトをレプリケートする] を選択してください。次に、レプリケート先オブジェクトを暗号化するための AWS KMS キーを選択します。

> **Note**
>
> レプリケート先バケットが別のアカウントにある場合は、レプリケート先アカウントが所有する AWS KMS カスタマーマネージドキーを指定します。デフォルトの Amazon S3 マネージドキー (aws/s3) は使用しないでください。デフォルトのキーを使用すると、レプリケート元アカウントが所有する Amazon S3 マネージドキーでオブジェクトが暗号化され、オブジェクトが別のアカウントと共有されるのを防ぎます。その結果、レプリケート先アカウントはレプリケート先バケット内のオブジェクトにアクセスできなくなります。

レプリケート先アカウントに属する AWS KMS キーを使用してレプリケート先オブジェクトを暗号化するには、レプリケート先アカウントによって KMS キーポリシーのレプリケーションロールに `kms:GenerateDataKey` および `kms:Encrypt` 権限を付与する必要があります。次の例のステートメントを KMS キーポリシーで使用するには、*`user input placeholders`* をユーザー自身の情報に置き換えます。

```
{
    "Sid": "AllowS3ReplicationSourceRoleToUseTheKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::123456789101:role/s3-replication-role"
    },
    "Action": ["kms:GenerateDataKey", "kms:Encrypt"],
    "Resource": "*"
```

```
}
```

AWS KMS キーポリシーの Resource ステートメントにアスタリスク (*) を使用すると、そのポリシーは KMS キーの使用権限をレプリケーションロールのみに付与します。このポリシーでは、レプリケーションロールの権限の昇格は許可していません。

デフォルトでは、KMS キーポリシーによって、キーに対するすべての権限をルートユーザーに付与します。この権限は、同じアカウントの他のユーザーに委任できます。ソース KMS キーポリシーに Deny ステートメントがない限り、IAM ポリシーを使用してレプリケーションロールにソース KMS キーへのアクセス権限を付与するだけで十分です。

> **Note**
>
> 特定の CIDR 範囲、VPC エンドポイント、S3 アクセスポイントへのアクセスを制限する KMS キーポリシーにより、レプリケーションが失敗する可能性があります。

レプリケート元またはレプリケート先の KMS キーのいずれかが、暗号化コンテキストに基づいてアクセス権限を付与する場合は、Amazon S3 バケットキーがバケットに対して有効になっていることを確認します。バケットで S3 バケットキーが有効になっている場合、暗号化コンテキストは次のようなバケットレベルのリソースである必要があります。

```
"kms:EncryptionContext:arn:aws:arn": [
     "arn:aws:s3:::SOURCE_BUCKET_NAME"
     ]
"kms:EncryptionContext:arn:aws:arn": [
     "arn:aws:s3:::DESTINATION_BUCKET_NAME"
     ]
```

KMS キーポリシーによって付与されるアクセス権限に加えて、レプリケート元アカウントは、レプリケーションロールの IAM ポリシーに次の最小限必要なアクセス権限を追加する必要があります。

```
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
```

```
    "Resource": [
        "SourceKmsKeyArn"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "kms:GenerateDataKey",
        "kms:Encrypt"
    ],
    "Resource": [
        "DestinationKmsKeyArn"
    ]
}
```

AWS KMS で暗号化されたオブジェクトをレプリケートする方法の詳細については、「暗号化オブジェクトのレプリケート」を参照してください。

- レプリケート先バケットが他の AWS アカウントによって所有されている場合、バケット所有者が、レプリケート元バケットの所有者に対してオブジェクトのレプリケーションを許可するバケットポリシーをレプリケート先バケットに設定していることを確認します。例については、「レプリケート元バケットとレプリケート先バケットが異なるアカウントによって所有されている場合での、レプリケーションの設定」を参照してください。
- 権限を検証してもオブジェクトがレプリケートされない場合は、次の場所に明示的な Deny ステートメントがないかを確認してください。
  - レプリケート先またはレプリケート先バケットポリシーにおける Deny ステートメント。バケットポリシーで次のいずれかのアクションでレプリケーションのロールへのアクセスを拒否した場合、レプリケーションは失敗します。

    レプリケート元のバケット

    ```
            "s3:GetReplicationConfiguration",
            "s3:ListBucket",
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
    ```

    レプリケート先バケット:

```
        "s3:ReplicateObject",
        "s3:ReplicateDelete",
        "s3:ReplicateTags"
```

  - IAM ロールにアタッチされた Deny ステートメントまたはアクセス許可の境界により、レプリケーションが失敗する可能性があります。
  - レプリケート元アカウントまたはレプリケート先アカウントのいずれかにアタッチされた AWS Organizations サービスコントロールポリシーの Deny ステートメントが原因で、レプリケーションが失敗する可能性があります。
- レプリケート先バケットにオブジェクトのレプリカが表示されない場合、次の問題でレプリケーションを妨げている可能性があります。
  - レプリケート元バケットのオブジェクト自体が別のレプリケーション設定によって作成されたレプリカである場合、Amazon S3 はそのオブジェクトをレプリケートしません。例えば、バケット A からバケット B へ、バケット B からバケット C へのレプリケーション設定を設定した場合、Amazon S3 はバケット B にあるオブジェクトレプリカをバケット C へレプリケートしません。
  - レプリケート元バケット所有者は、オブジェクトをアップロードするための許可を他の AWS アカウントに付与できます。デフォルトでは、レプリケート元バケット所有者は、他のアカウントによって作成されたオブジェクトに対するアクセス許可を持ちません。レプリケーション設定では、レプリケート元バケット所有者がアクセス許可を持つオブジェクトのみがレプリケートされます。レプリケート元バケット所有者は、作成されたオブジェクトに対する明示的なアクセス許可が必要であることを条件として、オブジェクトを作成する許可を他の AWS アカウントに与えることができます。ポリシーの例については、「バケット所有者はフルコントロール権限を持ちながら、オブジェクトをアップロードするためのクロスアカウントアクセス許可を付与する」を参照してください。
- レプリケーション設定に、特定のタグを持つオブジェクトのサブセットをレプリケートするためのルールを追加するとします。この場合、Amazon S3 がオブジェクトをレプリケートするには、オブジェクトの作成時に特定のタグキーと値を割り当てる必要があります。まず、オブジェクトを作成して、それから既存のオブジェクトにタグを追加した場合、Amazon S3 はそのオブジェクトをレプリケートしません。
- Amazon S3 イベント通知を使用すると、オブジェクトがレプリケート先の AWS リージョン にレプリケートされない場合に通知します。Amazon S3 イベント通知は、Amazon Simple Queue Service (Amazon SQS)、Amazon Simple Notification Service (Amazon SNS)、または AWS

Lambda を通じて使用できます。詳細については、「Amazon S3 イベント通知によるレプリケーション失敗イベントの受信」を参照してください。

また、Amazon S3 イベント通知を使用して、レプリケーションの失敗の理由を確認できます。失敗理由のリストを確認するには、「Amazon S3 レプリケーションの失敗の理由」を参照してください。

バッチレプリケーションエラー

レプリケート先バケットにレプリケートされないオブジェクトをトラブルシューティングするには、バケット、レプリケーションロール、バッチレプリケーションジョブの作成に使用した IAM ロールのさまざまなタイプのアクセス許可を確認します。また、パブリックアクセス設定とバケット所有権設定を必ず確認してください。

バッチレプリケーションの使用中に、次のいずれかのエラーが発生する可能性があります。

- バッチオペレーションの状態は失敗です。理由: ジョブレポートをレポートバケットに書き込めませんでした。

  このエラーは、バッチオペレーションジョブに使用される IAM ロールが、ジョブの作成時に指定された場所に完了レポートを保存できない場合に発生します。このエラーを解決するには、バッチオペレーションの完了レポートを保存するバケットの `PutObject` 権限が IAM ロールにあることを確認します。レポートは、レプリケート元バケットとは別のバケットに配信するのがベストプラクティスです。
- バッチオペレーションが失敗しました。失敗合計は 0 ではありません。

  このエラーは、実行中のバッチレプリケーションジョブにオブジェクト権限が不十分という問題がある場合に発生します。バッチレプリケーションジョブにレプリケーションルールを使用している場合は、レプリケーションに使用する IAM ロールに、レプリケート元またはレプリケート先のバケットのオブジェクトにアクセスする適切な権限があることを確認します。また、バッチレプリケーション完了レポートで、特定の Amazon S3 レプリケーション失敗の理由を確認できます。
- バッチジョブは正常に実行されましたが、レプリケート先バケットに必要なオブジェクトの数が一致しません。

  このエラーは、バッチレプリケーションジョブで提供されたマニフェストにリストアップされているオブジェクトと、ジョブの作成時に選択したフィルターが一致しない場合に発生します。また、レプリケート元バケット内のオブジェクトがどのレプリケーションルールにも一致せず、生成されたマニフェストに含まれていない場合にもこのメッセージが表示されることがあります。

サーバーのアクセスログ記録のトラブルシューティング

次のトピックは、Amazon S3 でログ記録された問題をトラブルシューティングするのに役立ちます。

トピック

ログ記録設定時のよくあるエラーメッセージ

AWS Command Line Interface (AWS CLI) と AWS SDK を使用してログ記録を有効にすると、次のようなエラーメッセージが表示されることがよくあります。

エラー: S3 ロケーションをまたがるロギングは許可されていません

送信先バケット (別名ターゲットバケット) がソースバケットとは別のリージョンに配置されている場合、Cross S3 location logging not allowed エラーが発生します。このエラーを解決するには、アクセスログを受信するように設定された送信先バケットがソースバケットと同じ AWS リージョン の AWS アカウント にあることを確認します。

エラー: ログを記録するバケットの所有者とターゲットバケットの所有者は同じである必要があります

サーバーアクセスのログ記録を有効にしている場合、指定された送信先バケットが別のアカウントに属している場合にこのエラーが発生します。このエラーを解決するには、送信先バケットがソースバケットと同じ AWS アカウント にあることを確認します。

> **Note**
>
> ソースバケットとは別のターゲットバケットを選択することをお勧めします。ソースバケットと送信先バケットが同じ場合、バケットに書き込まれるログに関する追加のログが作成され、ストレージ料金が増大する可能性があります。また、追加のログがあると、探している特定のログを見つけるのが難しくなることがあります。ログを管理しやすくするため、アクセスログは別のバケットに保存することをお勧めします。詳細については、「the section called “ログ配信を有効にするにはどうすればよいですか?”」を参照してください。

エラー: ログ記録のターゲットバケットが存在しません。

送信先バケットは、設定する前に存在している必要があります。このエラーは、送信先バケットが存在しないか、見つからないことを示しています。バケット名のスペルが正しいことを確認してから、もう一度試してください。

エラー: バケット所有者が強制するバケットにターゲット許可が付与されていません

このエラーは、送信先バケットが S3 オブジェクト所有権のバケット所有者の強制設定を使用していることを示しています。バケット所有者の強制設定は、送信先 (ターゲット) 権限をサポートしていません。詳細については、「ログ配信許可」を参照してください。

配信失敗のトラブルシューティング

サーバーアクセスのログ記録の問題を回避するには、次のベストプラクティスに従っていることを確認してください。

- S3 ログ配信グループに送信先バケットへの書き込みアクセス許可があること - S3 ログ配信グループは、送信先バケットにサーバーアクセスログを配信します。バケットポリシーまたはバケットアクセスコントロールリスト (ACL) を使用して、送信先バケットへの書き込みアクセスを付与できます。ただし、ACL の代わりにバケットポリシーを使用することをお勧めします。送信先バケットへの書き込みアクセスを許可する方法の詳細については、「ログ配信許可」を参照してください。

  > **Note**
  >
  > 送信先バケットが、オブジェクト所有権にバケット所有者の強制設定を使用している場合は、次の点に注意します。
  >
  > - ACL は無効になり、アクセス権限には影響しません。つまり、S3 ログ配信グループへのアクセスを許可すると、バケット ACL を更新できません。代わりに、ログサービスプリンシパルへのアクセスを許可するために、送信先バケットのバケットポリシーを更新する必要があります。
  > - `PutBucketLogging` 設定には、送信先権限を含めることできません。

- 送信先バケットのバケットポリシーがログへのアクセスを許可していること – ターゲットバケットのバケットポリシーを確認します。`"Effect": "Deny"` を含むステートメントをバケットポリシーで検索します。次に、`Deny` ステートメントがアクセスログのバケットへの書き込みを妨げていないことを確認します。
- S3 オブジェクトロックがターゲットバケットで有効になっていないこと – ターゲットバケットでオブジェクトロックが有効になっているかを確認します。オブジェクトロックにより、サーバーア

クセスログの配信をブロックします。オブジェクトロックが有効になっていない送信先バケットを選択する必要があります。

- 送信先バケットでデフォルトの暗号化が有効になっている場合、Amazon S3 マネージドキー (SSE-S3) が選択されていること – Amazon S3 マネージドキー (SSE-S3) でサーバー側の暗号化を使用する場合に限り、送信先バケットでデフォルトのバケット暗号化を使用できます。サーバーアクセスのログ記録の送信先バケットでは、AWS Key Management Service (AWS KMS) キー (SSE-KMS) によるデフォルトのサーバー側の暗号化はサポートされていません。デフォルトの暗号化の有効にする方法の詳細については、「デフォルトの暗号化の設定」を参照してください。
- ターゲットバケットでリクエスタ支払いが有効になっていないこと – サーバーアクセスのログ記録でのリクエスタ支払いバケットの使用はサポートされていません。サーバーアクセスログの配信を許可するには、ターゲットバケットのリクエスタ支払いオプションを無効にします。
- AWS Organizations サービスコントロールポリシーを確認する - AWS Organizations を使用している場合は、サービスコントロールポリシーをチェックして、Amazon S3 へのアクセスが許可されていることを確認します。サービスコントロールポリシーは、影響を受けるアカウントのアクセス許可の上限を指定します。サービスコントロールポリシーに `"Effect": "Deny"` を含むステートメントを検索し、Deny ステートメントがバケットへのアクセスログの書き込みを妨げていないことを確認します。詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー (SCPs)」を参照してください。
- 最新のロギング設定の変更が有効になるまで時間がかかること - サーバーアクセスのログ記録を最初に有効にした場合や、ログの送信先バケットを変更した場合は、すべてが有効になるまでに時間がかかります。すべてのリクエストが正しく記録され、配信されるまでに 1 時間以上かかる場合があります。

  ログ配信の失敗を確認するには、Amazon CloudWatch でリクエストメトリクスを有効にします。ログが数時間以内に配信されない場合は、ログ配信の失敗を示す `4xxErrors` メトリクスを探してください。リクエストメトリクスの有効化の詳細については、「the section called “すべてのオブジェクトに対するメトリクス設定の作成”」を参照してください。

バージョニングのトラブルシューティング

次のトピックは Amazon S3 のバージョニングのよくある問題をトラブルシューティングするのに役立ちます。

トピック

- バージョニングが有効なバケットで、誤って削除されたオブジェクトを復元する
- バージョニングされたオブジェクトを完全に削除する

- バケットのバージョニングを有効にした後、パフォーマンスが低下している

バージョニングが有効なバケットで、誤って削除されたオブジェクトを復元する

一般に、S3 バケットからオブジェクトバージョンが削除された場合、それらを Amazon S3 で復元する方法はありません。ただし、S3 バケットで S3 バージョニングを有効にしている場合、バージョン ID を指定しない DELETE リクエストでは、オブジェクトを完全に削除できません。代わりに、削除マーカーがプレースホルダーとして追加されます。この削除マーカーはオブジェクトの最新バージョンになります。

削除したオブジェクトが完全に削除されているのか、一時的に削除されているのか (その場所に削除マーカーが表示されているか) を確認するには、次の手順に従います。

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. [バケット] リストで、オブジェクトが含まれるバケットの名前を選択します。
4. [オブジェクト]リストで、検索バーの右側にある[バージョンの表示]トグルをオンにして、検索バーで削除されたオブジェクトを検索します。このトグルは、バケットで以前にバージョニングが有効だった場合に限り使用できます。

   S3 インベントリを使用して、削除されたオブジェクトを検索することもできます。
5. [バージョンの表示] の切り替えや、インベントリレポートの作成の後にオブジェクトが見つからず、オブジェクトの削除マーカーも見つからない場合、完全に削除され、オブジェクトを復元することはできません。

また、AWS Command Line Interface (AWS CLI) の HeadObject API オペレーションを使用して、削除されたオブジェクトの状態を確認できます。それには、次の head-object コマンドを使用して、*user input placeholders* を自分の情報に置き換えます。

```
aws s3api head-object --bucket example-s3-bucket --key index.html
```

最新バージョンが削除マーカーであるバージョニングされたオブジェクトに対して head-object コマンドを実行すると、404 Not Found エラーが表示されます。例:

HeadObject オペレーションの呼び出し時にエラーが発生しました (404): Not Found

バージョニングされたオブジェクトに対して head-object コマンドを実行し、オブジェクトのバージョン ID を指定すると、Amazon S3 はオブジェクトのメタデータを取得し、オブジェクトがまだ存在し、完全に削除されていないことを確認します。

```
aws s3api head-object --bucket example-s3-bucket --key index.html --
version-id versionID
```

```
{
"AcceptRanges": "bytes",
"ContentType": "text/html",
"LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
"ContentLength": 77,
"VersionId": "Zg5HyL7m.eZU9iM7AVlJkrqAiE.0UG4q",
"ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
"Metadata": {}
}
```

オブジェクトが見つかり、最新バージョンが削除マーカーの場合、オブジェクトの以前のバージョンがまだ存在します。削除マーカーはオブジェクトの最新バージョンであるため、削除マーカーを削除することでオブジェクトを復元できます。

削除マーカーを完全に削除すると、2 番目に新しいバージョンのオブジェクトが最新のバージョンのオブジェクトになり、オブジェクトを再び使用できるようになります。オブジェクトの復元方法を視覚的に説明するには、「削除マーカーの削除」を参照してください。

特定のバージョンのオブジェクトを削除するには、バケット所有者になる必要があります。削除マーカーを完全に削除するには、DeleteObject リクエストにそのバージョン ID を含める必要があります。それには、次のコマンドを使用して、*user input placeholders* を自分の情報に置き換えます。

```
aws s3api delete-object --bucket example-s3-bucket --key index.html --
version-id versionID
```

delete-object コマンドの詳細については、AWS CLI コマンドリファレンスの「delete-object」を参照してください。削除マーカーの完全削除の詳細については、「削除マーカーの管理」を参照してください。

バージョニングされたオブジェクトを完全に削除する

バージョニングが有効なバケットでは、バージョン ID のない DELETE リクエストではオブジェクトを完全に削除することはできません。このようなリクエストでは、代わりに削除マーカーが挿入されます。

バージョニングされたオブジェクトを完全に削除するには、以下の方法から選択できます。

- S3 ライフサイクルルールを作成して、最新ではないバージョンを完全に削除します。最新ではないバージョンのオブジェクトを完全に削除するには、[現在のバージョンではないオブジェクトを完全に削除する]を選択し、[オブジェクトが最新バージョンでなくなってからの日数]に日数を入力します。オプションで、[Number of newer versions to retain] (保持する新しいバージョンの数) に値を入力して、保持する新しいバージョンの数を指定できます。このルールの作成については、「S3 ライフサイクル設定の指定」を参照してください。
- DELETE リクエストにバージョン ID を含め、指定されたバージョンを削除します。詳細については、「バージョニングされたオブジェクトの完全削除方法」を参照してください。
- 現在のバージョンを期限切れにするライフサイクルルールを作成します。オブジェクトの現在のバージョンを期限切れにするには、[オブジェクトの現行バージョンを期限切れにする]を選択し、[オブジェクトの作成後の日数]に日数を入力します。このライフサイクルルールの作成については、「S3 ライフサイクル設定の指定」を参照してください。
- バージョニングされたオブジェクトと削除マーカーをすべて完全に削除するには、2 つのライフサイクルルールを作成します。1 つは現在のバージョンを期限切れにして古いバージョンのオブジェクトを完全に削除するルールで、もう 1 つは期限切れのオブジェクト削除マーカーを削除するルールです。

バージョニングが有効なバケットでは、バージョン ID を指定しない DELETE リクエストでは、NULL バージョン ID を持つオブジェクトのみを削除できます。バージョニングが有効の場合にオブジェクトがアップロードされた場合、バージョン ID を指定しない DELETE リクエストにより、そのオブジェクトの削除マーカーを作成します。

> **Note**
>
> S3 オブジェクトロックが有効なバケットでは、保護されたオブジェクトバージョン ID を含む DELETE オブジェクトリクエストにより、403 Access Denied エラーが発生します。バージョン ID のない DELETE オブジェクトリクエストでは、200 OK レスポンスでオブジェクトの最新バージョンとして削除マーカーが追加されます。オブジェクトロックで保護されているオブジェクトは、その保存期間とリーガルホールドが解除されるまで完全に削除できませ

ん。詳細については、「the section called “S3 オブジェクトロックの仕組み”」を参照してください。

バケットのバージョニングを有効にした後、パフォーマンスが低下している

削除マーカーやバージョニングされたオブジェクトが多すぎる場合や、ベストプラクティスに従わない場合、バージョニングが有効なバケットでパフォーマンスが低下する可能性があります。

削除マーカーが多すぎる

バケットでバージョニングを有効にすると、オブジェクトに対してバージョン ID を持たない DELETE リクエストを行うと、一意のバージョン ID の削除マーカーが作成されます。現在のバージョンのオブジェクトを期限切れにするルールを含むライフサイクル設定では、すべてのオブジェクトに一意のバージョン ID を持つ削除マーカーが追加されます。削除マーカーが多すぎると、バケットのパフォーマンスが低下する可能性があります。

バケットでバージョニングが停止すると、Amazon S3 では新しく作成されたオブジェクトに対するバージョン ID を NULL としてマークします。バージョニングが停止されたバケットで期限切れアクションを行うと、Amazon S3 がバージョン ID として NULL を使用する削除マーカーを作成します。バージョニングが停止されたバケットでは、すべての削除リクエストに対して NULL 削除マーカーが作成されます。これらの NULL 削除マーカーは、すべてのオブジェクトバージョンが削除され、単一の削除マーカーだけが残っている場合、期限切れオブジェクト削除マーカーとも呼ばれます。NULL 削除マーカーが多すぎると、バケットのパフォーマンスが低下します。

バージョニングされたオブジェクトが多すぎる

バージョニングが有効なバケットに数百万のバージョンのオブジェクトが含まれている場合、503 Service Unavailable エラーが増える可能性があります。バージョニングを有効にしたバケットへの PUT または DELETE オブジェクトリクエストに対して受信される HTTP 503 Service Unavailable レスポンスの数が著しく増加した場合、バケットに数百万のバージョンが存在するオブジェクトがある可能性があります。何百万ものバージョンのオブジェクトがある場合、Amazon S3 はバケットへのリクエストを自動的に制限します。リクエストのスロットリングによって、過剰なリクエストトラフィックからバケットを保護しますが、同じバケットに対して行われた他のリクエストに支障がある可能性があります。

数百万のバージョンがあるオブジェクトを確認するには、S3 インベントリを使用します。S3 インベントリは、バケット内のオブジェクトのフラットファイルリストを提供するレポートを生成します。詳細については、「Amazon S3 インベントリ」を参照してください。

バケット内でバージョニングされたオブジェクト数が多いかどうかを確認するには、S3 Storage Lens メトリクスを使用して、現在のバージョンのオブジェクト数、最新ではないバージョンのオブジェクト数、削除マーカーのオブジェクト数を表示します。Storage Lens メトリクスの詳細については、「Amazon S3 Storage Lens のメトリクスに関する用語集」を参照してください。

Amazon S3 チームでは、同じオブジェクトを繰り返し上書きし、そのオブジェクトに対して数百万のバージョンを作成する可能性のあるアプリケーションを調査して、アプリケーションが意図どおり動作しているかどうか調査することをお客様にお勧めしています。例えば、アプリケーションが毎分同じオブジェクトを 1 週間上書きすると、1 万を超えるバージョンが作成されることになります。各オブジェクトに保存するバージョンは 10 万未満にすることをお勧めします。単一または複数の Amazon S3 オブジェクトに対して数百万のバージョンを必要とするユースケースがある場合の最適なソリューションを決定するには、AWS Support チームにご相談ください。

ベストプラクティス

バージョニングに関連したパフォーマンス低下の問題を防ぐため、次のベストプラクティスを採用することをお勧めします。

- ライフサイクルルールを有効にして、以前のバージョンのオブジェクトを期限切れにする。例えば、オブジェクトが最新ではない状態になってから 30 日後に、最新ではないバージョンを期限切れにするライフサイクルルールを作成できます。すべてを削除する必要がない場合は、最新ではないバージョンを複数保持することもできます。詳細については、「S3 ライフサイクル設定の指定」を参照してください。
- ライフサイクルルールを有効にして、バケットに関連するデータオブジェクトがない期限切れのオブジェクト削除マーカーを削除します。詳細については、「期限切れオブジェクト削除マーカーの削除」を参照してください。

Amazon S3 のパフォーマンス最適化に関するその他のベストプラクティスについては、「ベストプラクティスのデザインパターン」を参照してください。

AWS Support の Amazon S3 リクエスト ID の取得

Amazon S3 でエラーまたは予期しない動作が発生したために、AWS Support に問い合わせるときは、失敗したアクションに関連するリクエスト ID を提供する必要があります。 AWS Support はこれらのリクエスト ID を使用して、発生している問題を解決します。

リクエスト ID はペアになっていて、Amazon S3 が処理するすべての応答で返され (エラーの応答を含む)、詳細なログを通じてアクセスできます。リクエスト ID を取得するための一般的な方法がいくつかあります。例えば、S3 アクセスログや AWS CloudTrail イベントまたはデータイベントなどがあります。

これらのログを回復したら、2 つの値をコピーして記録しておきます。この値は、 連絡するとき必要になりますAWS Support AWS Support へのお問い合わせの詳細については、AWS Support ドキュメントの「AWS へのお問い合わせ」を参照してください。

HTTP を使用したリクエスト ID の取得

ターゲットアプリケーションに到達する前に HTTP リクエストのビットを記録することで、リクエスト ID、`x-amz-request-id`、および `x-amz-id-2` を取得することができます。HTTP リクエストの詳細なログを復元するために使用できる、さまざまなサードパーティー製ツールがあります。別の Amazon S3 HTTP リクエストを送信するにあたって、信頼しているツールを選択して実行し、Amazon S3 トラフィックが流れるポートをリッスンします。

HTTP リクエストの場合、リクエスト ID のペアは次のようになります。

```
x-amz-request-id: 79104EXAMPLEB723
x-amz-id-2: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km
```

> **Note**
>
> HTTPS リクエストで暗号化され、ほとんどのパケットキャプチャで非表示になります。

ウェブブラウザを使用したオブジェクトリクエスト ID の取得

ほとんどのウェブブラウザには、リクエストヘッダーを確認できる開発者ツールがあります。

ウェブブラウザに基づくリクエストでエラーが返される場合、リクエスト ID のペアは次の例のようになります。

```
<Error><Code>AccessDenied</Code><Message>Access Denied</Message>
<RequestId>79104EXAMPLEB723</RequestId><HostId>IOWQ4fDEXAMPLEQM
+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km</HostId></Error>
```

成功したリクエストからリクエスト ID のペアを取得するには、開発者ツールを使用して、HTTP レスポンスヘッダーを表示する必要があります。特定のブラウザ用のデベロッパーツールの詳細については、AWS re:Post の「Amazon S3 のトラブルシューティング - S3 リクエスト ID の復旧方法」を参照してください。

AWS SDK を使用したリクエスト ID の取得

以下のセクションでは、AWS SDK を使用したログ記録の設定に関する情報を示します。すべてのリクエストと応答で詳細ログを有効にすることができますが、大きなリクエストまたは応答はアプリケーションで大幅な遅延を発生させる可能性があるため、本稼働システムでログ記録を有効にすることはお勧めしません。

AWS SDK リクエストの場合、リクエスト ID のペアは次の例のようになります。

```
Status Code: 403, AWS Service: Amazon S3, AWS Request ID: 79104EXAMPLEB723
AWS Error Code: AccessDenied  AWS Error Message: Access Denied
S3 Extended Request ID: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK
+Jd1vEXAMPLEa3Km
```

SDK for Go を使用したリクエスト ID の取得

ログ記録の設定は、SDK for Go を使用して行うことができます。詳細については、「SDK for Go V2 デベロッパーガイド」の「Response metadata」を参照してください。

SDK for PHP を使用したリクエスト ID の取得

ログ記録の設定は、PHP を使用して行うことができます。詳細については、AWS SDK for PHP デベロッパーガイドの「送信されるデータを表示する方法」を参照してください。

SDK for Java を使用したリクエスト ID の取得

特定のリクエストまたは応答のログ記録を有効にして、関連するヘッダーのみを取得して返すことができます。これを行うには、`com.amazonaws.services.s3.S3ResponseMetadata` クラスをインポートします。その後、実際のリクエストを実行する前に変数にリクエストを保存できます。記録されたリクエストまたは応答を取得す

るには、getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID() を呼び出します。

Example

```
PutObjectRequest req = new PutObjectRequest(bucketName, key, createSampleFile());
s3.putObject(req);
S3ResponseMetadata md = s3.getCachedResponseMetadata(req);
System.out.println("Host ID: " + md.getHostId() + " RequestID: " + md.getRequestId());
```

または、Java のすべてのリクエストと応答の詳細ログ記録を使用できます。詳細については、AWS SDK for Java デベロッパーガイドで「詳細なワイヤログ記録」を参照してください。

AWS SDK for .NET を使用したリクエスト ID の取得

組み込みの System.Diagnostics ログ作成ツールを使用して、AWS SDK for .NET のログ記録を設定できます。詳細については、AWS デベロッパーブログの投稿である「AWS SDK for .NET によるログ記録」を参照してください。

> **Note**
>
> デフォルトでは、返されたログにはエラー情報のみが含まれます。リクエスト ID を取得するには、設定ファイルに AWSLogMetrics (およびオプションで AWSResponseLogging) を追加する必要があります。

SDK for Python (Boto3) を使用したリクエスト ID の取得

AWS SDK for Python (Boto3) を使用すると、特定の応答をログに記録できます。この機能を使用して、関連するヘッダーのみを取得できます。次のコードは、レスポンスの一部をファイルに記録する方法を示しています。

```
import logging
import boto3
logging.basicConfig(filename='logfile.txt', level=logging.INFO)
logger = logging.getLogger(__name__)
s3 = boto3.resource('s3')
response = s3.Bucket(bucket_name).Object(object_key).put()
logger.info("HTTPStatusCode: %s", response['ResponseMetadata']['HTTPStatusCode'])
logger.info("RequestId: %s", response['ResponseMetadata']['RequestId'])
logger.info("HostId: %s", response['ResponseMetadata']['HostId'])
```

```
logger.info("Date: %s", response['ResponseMetadata']['HTTPHeaders']['date'])
```

例外を発生時にキャッチし、関連情報をログに記録することもできます。詳細については、AWSSDK for Python (Boto) API リファレンスの「エラー応答から有用な情報を識別する」を参照してください。

さらに、次のコードを使用して、詳細なデバッグログを出力するように Boto3 を設定できます。

```
import boto3
boto3.set_stream_logger('', logging.DEBUG)
```

詳細については、AWS SDK for Python (Boto) API リファレンスの「set_stream_logger」を参照してください。

SDK for Ruby を使用したリクエスト ID の取得

SDK for Ruby バージョン 1、2、または 3 を使用して、リクエスト ID を取得できます。

- SDK for Ruby - バージョン 1 の使用 – 次のコードで、HTTP ワイヤログをグローバルに有効にすることができます。

  ```
  s3 = AWS::S3.new(:logger => Logger.new($stdout), :http_wire_trace => true)
  ```

- SDK for Ruby - バージョン 2 またはバージョン 3 の使用 – 次のコードで、HTTP ワイヤログをグローバルに有効にすることができます。

  ```
  s3 = Aws::S3::Client.new(:logger => Logger.new($stdout), :http_wire_trace => true)
  ```

AWS クライアントからワイヤ情報を取得するためのヒントについては、「デバッグのヒント: ワイヤトレース情報をクライアントから取得」を参照してください。

AWS CLI を使用したリクエスト ID の取得

AWS Command Line Interface(AWS CLI) を使用してリクエスト ID を取得するには、`--debug` を コマンドに追加してください。

Windows PowerShell を使用してリクエスト ID を取得する

Windows PowerShell を使用してログを回復する方法については、.NET 開発ブログの投稿の「AWS Tools for Windows PowerShell における応答ログ」を参照してください。

AWS CloudTrail データイベントを使用してリクエスト ID を取得する

S3 オブジェクトレベルの API オペレーションをログに記録するように CloudTrail データイベントで設定された Amazon S3 バケットは、Amazon S3 でユーザー、ロール、または AWS サービスによって実行されるアクションに関する詳細情報を提供します。Athena で CloudTrail イベントをクエリすることで、S3 リクエスト ID を特定できます。

S3 サーバーアクセスログ記録を使用してリクエスト ID を取得する

S3 サーバーアクセスログ用に設定された Amazon S3 バケットは、バケットに対して行われた各リクエストの詳細な記録を提供します。Athena を使用してサーバーアクセスログをクエリすることで、S3 リクエスト ID を特定できます。

ドキュメント履歴

- 現行 API バージョン: 2006-03-01

次の表に、Amazon Simple Storage Service API リファレンスおよび Amazon S3 ユーザーガイドの各リリースにおける重要な変更点が記されています。このドキュメントの更新に関する通知については、RSS フィードにサブスクライブできます。

変更	説明	日付
Amazon S3 インベントリが s3:InventoryAccessibleOptionalFields 条件キーをサポート	Amazon S3 インベントリは s3:InventoryAccessibleOptionalFields 条件キーをサポートしており、ユーザーがレポートにオプションのメタデータフィールドを含めることができるかどうかを制御します。詳細については、「S3 インベントリレポート設定の作成を制御する」を参照してください。	2024 年 2 月 20 日
S3 on Outposts の IPv6 サポート	S3 on Outposts デュアルスタックエンドポイント経由で IPv6 を使用して、S3 on Outposts バケットにアクセスできるようになりました。S3 on Outposts での IPv6 サポートにより、S3 on Outposts バケットとコントロールプレーンリソースを IPv6 ネットワーク経由で管理できます。	2024 年 1 月 16 日
新しい高パフォーマンスのシングルゾーン Amazon S3 ス	Amazon S3 Express One Zone は、最もレイテンシーの影響を受けやすいアプリ	2023 年 11 月 28 日

トレージクラス – S3 Express One Zone	ケーションに 1 桁のミリ秒単位で一貫したデータアクセスを提供することを目的として構築された、高パフォーマンスのシングルアベイラビリティーゾーンの Amazon S3 ストレージ クラスです。詳細については、「S3 Express One Zone」を参照してください。	
Mountpoint for Amazon S3 での S3 Express One Zone のサポートの追加	S3 Express One Zone のディレクトリバケットを Mountpoint を使用してマウントできるようになりました。	2023年11月28日
Lambda 呼び出しスキーマのバージョン	Amazon S3 バッチオペレーションでは、ディレクトリバケットで動作するバッチオペレーションジョブで使用するための新しい Lambda 呼び出しスキーマバージョンが導入されました。詳細については、「ディレクトリバケットでの Lambda と Amazon S3 のバッチオペレーションの使用」を参照してください。	2023年11月28日

ディレクトリバケットのインポートアクション	Amazon S3 でインポートアクションが提供されるようになりました。インポートは、オブジェクトを汎用バケットからディレクトリバケットにコピーする Amazon S3 バッチオペレーションジョブを作成するための効率的な方法です。詳細については、「ディレクトリバケットへのオブジェクトのインポート」を参照してください。	2023年11月28日
S3 Access Grants で S3 アクセスを管理する	Amazon S3 Access Grants を使用すると、AWS Identity and Access Management (IAM) プリンシパルのデータアクセス許可を大規模に管理できます。また、Azure AD などの企業ディレクトリからのディレクトリアイデンティティも管理できます。最小権限の S3 アクセス権限を強制し、ビジネスニーズに基づいてそれらのアクセス権限を簡単にスケーリングできるようになりました。詳細については、「S3 Access Grants でのアクセス管理」を参照してください。	2023 年 11 月 26 日
Mountpoint for Amazon S3 バッチオペレーション	Mountpoint で、繰り返しアクセスするデータのキャッシュを設定できるようになりました。	2023年11月22日

Amazon S3 バッチオペレーションのマニフェスト生成の拡張	ジョブを作成する際に指定するオブジェクトフィルタ条件に基づいたマニフェストの自動生成を Amazon S3 Batch オペレーションに指示できるようになりました。このオプションは、Amazon S3 コンソールで作成したバッチレプリケーションジョブ、または AWS CLI、AWS、Amazon S3 REST API を使用して作成した任意のジョブタイプで使用できます。詳細については、「Amazon S3 バッチオペレーションジョブの作成」を参照してください。	2023年11月22日
既存の Amazon S3 バケットでの Object Lock 設定の追加	既存の Amazon S3 バケットで Object Lock を有効にできるようになりました。新規または既存のバケットにリーガルホールドと保持期間を設定できます。詳細については、「オブジェクトロックの使用」を参照してください。	2023年11月20日
プレフィックスの S3 Storage Lens リクエストメトリクス	S3 Storage Lens は、Amazon S3 バケット内のプレフィックスに対してリクエストメトリクスを導入しています。詳細については、「メトリクスカテゴリ」を参照してください。	2023 年 11 月 17 日

Amazon S3 ストレージレンズグループ	S3 Storage Lens には、オブジェクトメタデータに基づいてオブジェクト用のカスタム定義フィルタである Storage Lens グループが導入されています。詳細については、「Amazon S3 ストレージレンズグループの使用」を参照してください。	2023 年 11 月 15 日
新しい IAM ポリシー	S3 on Outposts では、ネットワークリソースの管理に役立つ、サービスにリンクされたロール `AWSServiceRoleForS3OnOutposts` が導入されました。詳細については、「S3 on Outposts でのサービスにリンクされたロールの使用」を参照してください。	2023 年 10 月 3 日
Amazon S3 は、削除マーカーの `Last-Modified` 時間を提供します。	Amazon S3 は、S3 `Head` オペレーションと `Get` API オペレーションのレスポンスヘッダーに、削除マーカーの `Last-Modified` 時間を提供します。詳細については、「削除マーカーの使用」を参照してください。	2023 年 9 月 27 日
AWS マネージドポリシーに対する Amazon S3 の最新情報	Amazon S3 が `s3:Describe*` 許可を `AmazonS3ReadOnlyAccess` に追加しました。詳細については、「Amazon S3 の AWS マネージドポリシー」を参照してください。	2023 年 8 月 11 日

S3 バッチオペレーションによる標準復元リクエストの開始時間が短縮されました	S3 バッチオペレーションによる復元リクエストの標準取得を数分で開始できるようになりました。詳細については、「アーカイブの取り出しオプション」を参照してください。	2023 年 8 月 9 日
Amazon S3 バケットをローカルファイルシステムとしてマウントするための、高スループットのオープンソースファイルクライアントである、Mountpoint を追加しました。	Mountpoint を使うと、アプリケーションはファイルオペレーションを通じて Amazon S3 に保存されているオブジェクトにアクセスできるため、アプリケーションはファイルインターフェイスを通じて Amazon S3 の伸縮自在なストレージとスループットにアクセスできます。	2023 年 8 月 9 日
AWS Key Management Service キーによる二層式サーバー側の暗号化 (DSSE-KMS)	AWS Key Management Service (AWS KMS) キーによる二層式サーバー側の暗号化 (DSSE-KMS) を使用すると、オブジェクトが Amazon S3 にアップロードされるときに 2 つの暗号化レイヤーが適用されます。詳細については、「AWS KMS キーによる二層式サーバー側の暗号化の使用」を参照してください。	2023 年 6 月 13 日

Amazon S3 は、すべての新しいバケットについて、S3 ブロックパブリックアクセスを有効にし、S3 アクセスコントロールリスト (ACL) を無効にします。	Amazon S3 は、すべての AWS リージョンのすべての新しい S3 バケットについて、S3 ブロックパブリックアクセスを自動的に有効にし、S3 アクセスコントロールリスト (ACL) を無効にするようになりました。詳細については、「Amazon S3 ストレージへのパブリックアクセスをブロックする」と「オブジェクトの所有権の制御とバケットの ACL の無効化」を参照してください。	2023 年 4 月 27 日
S3 レプリケーションオペレーション失敗メトリクス	Amazon S3 には、S3 レプリケーションの失敗をモニタリングするための新しい Amazon CloudWatch メトリクスが追加されています。詳細については、「レプリケーションメトリクスによる進捗状況のモニタリング」を参照してください。	2023 年 4 月 5 日
プライベート DNS	Amazon S3 の AWS PrivateLink がプライベート DNS をサポートするようになりました。詳細については、「プライベート DNS」を参照してください。	2023 年 3 月 14 日

Amazon S3 コンソールでのクロスアカウントアクセスポイントのサポート	Amazon S3 では、Amazon S3 コンソールによるクロスアカウントのアクセスポイントの作成がサポートされるようになりました。詳細については、「アクセスポイントの作成」を参照してください。	2023 年 3 月 14 日
Amazon S3 on Outposts は S3 Replication on Outposts をサポートします	ローカル S3 レプリケーションを使用すると、オブジェクトを単一の Outposts レプリケート先バケットまたは複数のレプリケート先バケットに自動的に複製できます。レプリケート先バケットは、異なる AWS Outposts でも、レプリケート元バケットと同じ Outposts 内でも配置することができます。詳細については、「S3 on Outposts のオブジェクトのレプリケート」を参照してください。	2023 年 3 月 14 日

Amazon S3 Object Lambda アクセスポイントのエイリアス	Object Lambda アクセスポイントを作成すると、Amazon S3 は Object Lambda アクセスポイントの固有のエイリアスを自動的に生成します。アクセスポイントのデータプレーンオペレーションのリクエストで、Amazon S3 バケット名や Object Lambda アクセスポイント Amazon Resource Name (ARN) の代わりに、このエイリアスを使用することができます。詳細については、「Amazon S3 Object Lambda アクセスポイントの使用」を参照してください。	2023 年 3 月 14 日
Amazon S3 マルチリージョンアクセスポイントのクロスアカウントサポート	Amazon S3 では、Amazon S3 コンソールによるクロスアカウントのマルチリージョンアクセスポイントの作成がサポートされるようになりました。詳細については、「マルチリージョンアクセスポイントの作成」を参照してください。	2023 年 3 月 14 日

クロスアカウントアクセスポイント	Amazon S3 は、クロスアカウントアクセスポイントの作成をサポートしています。AWS Command Line Interface (AWS CLI) または REST API `CreateAccessPoint` オペレーションを使用してクロスアカウントアクセスポイントを作成できます。詳細については、「アクセスポイントの作成」を参照してください。	2022 年 11 月 30 日
Amazon S3 で Amazon S3 マルチリージョンアクセスポイントのフェイルオーバーコントロールをサポート	Amazon S3 で、Amazon S3 マルチリージョンアクセスポイントのフェイルオーバーコントロールが導入されました。これらのコントロールにより、Amazon S3 マルチリージョンアクセスポイントを経由する S3 データアクセスリクエストトラフィックを数分以内に別の AWS リージョン に移行し、可用性の高いアプリケーションをテストおよび構築することができます。詳細については、「Amazon S3 Multi-Region Access Point failover controls」(Amazon S3 マルチリージョンアクセスポイントのフェイルオーバーコントロール) を参照してください。	2022 年 11 月 28 日

Amazon S3 ストレージレンズの 34 の新しいメトリクスで組織全体の可視性を向上	S3 ストレージレンズは、より深いコスト最適化の機会を発見し、データ保護のベストプラクティスを特定し、アプリケーションワークフローのパフォーマンスを向上させるための 34 の追加メトリクスを導入しました。詳細については、「S3 Storage Lens metrics」(S3 ストレージレンズメトリクス) を参照してください。	2022 年 11 月 17 日
Amazon S3 で、S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive のより高い復元リクエストのレートをサポート	Amazon S3 は、S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive ストレージクラスで、AWS アカウント ごとに、1 秒あたり最大 1,000 件のトランザクションの復元リクエストをサポートします。	2022 年 11 月 15 日

Amazon S3 on Outposts は、追加の S3 ライフサイクルアクションとフィルタをサポート	S3 on Outposts は、キャパシティ管理を最適化するための追加の S3 ライフサイクルルールをサポートしています。オブジェクトが古くなったり、新しいバージョンに置き換えられたりしたときに期限切れにすることができます。プレフィックス、オブジェクトタグ、オブジェクトサイズでフィルタリングすることで、バケット全体またはバケット内のオブジェクトのサブセットに対してライフサイクルルールを作成することができます。ライフサイクル設定の管理の詳細については、「Amazon S3 on Outposts バケットのライフサイクル設定を作成および管理する」を参照してください。	2022 年 11 月 2 日
SSE-C オブジェクトに対する S3 レプリケーションのサポート	カスタマー提供キーでサーバーサイド暗号化を使用して作成されたオブジェクトをレプリケートすることができます。暗号化されたオブジェクトのレプリケートについて詳しくは、「サーバー側の暗号化 (SSE-C、SSE-S3、SSE-KMS) で作成されたオブジェクトをレプリケートする」を参照してください。	2022 年 10 月 24 日

Amazon S3 on Outposts でアクセスポイントのエイリアスをサポート	S3 on Outposts では、Outposts バケット内の任意のオブジェクトにアクセスするにはアクセスポイントを使用する必要があります。バケットのアクセスポイントを作成するたびに、S3 on Outposts によってアクセスポイントのエイリアスが自動的に生成されます。このアクセスポイントエイリアスは、あらゆるデータプレーンオペレーションでアクセスポイント ARN の代わりに使用できます。詳細については、「Using a bucket-style alias for your S3 on Outposts bucket access point」(S3 on Outposts バケットアクセスポイントにバケット形式のエイリアスを使用する) を参照してください。	2022 年 10 月 21 日
S3 Object Lambda で `HeadObject` 、`ListObjects` 、および `ListObjectsV2` オペレーションをサポート	カスタムコードを使用して、標準の S3 `GET`、`LIST`、`HEAD` リクエストによって返されるデータを変更し、行のフィルタリング、画像の動的なサイズ変更、機密データの編集などを行うことができるようになりました。詳細については、「S3 Object Lambda を使用したオブジェクトの変換」を参照してください。	2022 年 10 月 4 日

Amazon S3 on Outposts で S3 バージョニングをサポート	S3 バージョニングを有効にすると、オブジェクトの複数の異なるコピーが同じバケット内に保存されます。S3 バージョニングを使用すると、Outposts バケットに保存されたあらゆるオブジェクトのあらゆるバージョンを保存、取得、復元できます。S3 バージョニングによって、意図しないユーザーアクションやアプリケーション障害から復旧できます。詳細については、「Managing S3 Versioning for your S3 on Outposts bucket」(S3 on Outposts バケットの S3 バージョニングの管理) を参照してください。	2022 年 9 月 21 日
Amazon S3 用 AWS Backup	AWS Backup は、Amazon S3 のデータを保護するための中央バックアップポリシーを定義するために使用できる、フルマネージド型のポリシーベースのサービスです。詳細については、「Amazon S3 用の AWS Backup の使用」を参照してください。	2022 年 2 月 18 日

S3 バッチレプリケーションを使用して既存のオブジェクトをレプリケートする	S3 バッチレプリケーションを使用すると、レプリケーション設定が実行される前に存在していたオブジェクトをレプリケートできます。既存のオブジェクトのレプリケートは、バッチオペレーションジョブを使用して行います。S3 バッチレプリケーションは、Amazon S3 バケット間で新しいオブジェクトを継続的かつ自動的にコピーするライブレプリケーションとは異なります。詳細については、「S3 バッチレプリケーションを使用した既存のオブジェクトをレプリケートする」を参照してください。	2022 年 2 月 8 日
S3 Glacier Flexible Retrieval の名前変更	Glacier ストレージクラスが S3 Glacier Flexible Retrieval に変更されました。この変更は API に影響を与えません。	2021 年 11 月 30 日

ACL を無効にする新しい S3 オブジェクトの所有権の設定	オブジェクト所有権のバケット所有者強制設定を適用すると、バケットとその中のオブジェクトの ACL を無効にして、バケット内のすべてのオブジェクトの所有権を取得できます。バケット所有者の強制設定により、Amazon S3 に保存されているデータのアクセス管理が簡単になります。詳細については、バケットのオブジェクト所有権のコントロールと ACL の無効化を参照してください。	2021 年 11 月 30 日
新しい S3 Intelligent-Tiering ストレージクラス	S3 Intelligent-Tiering のアーカイブインスタントアクセスは、S3 Intelligent-Tiering の下位にあるストレージクラスです。詳細については、「S3 Intelligent-Tiering の仕組み」を参照してください。	2021 年 11 月 30 日
新しい S3 Glacier Instant Retrieval ストレージクラス	S3 Glacier Instant Retrieval ストレージクラスに、オブジェクトを配置できるようになりました。このストレージクラスの詳細については、Amazon S3 ストレージクラス を参照してください。	2021 年 11 月 30 日

Amazon S3 用 AWS Backup プレビュー	AWS Backup は、Amazon S3 のデータを保護するための中央バックアップポリシーを定義するために使用できる、フルマネージド型のポリシーベースのサービスです。詳細については、Amazon S3 AWS Backup の使用方法を参照してください。	2021 年 11 月 30 日
Amazon S3 用 AWS Identity and Access Management Access Analyzer	IAM Access Analyzer は、IAM ポリシーの文法 および ベストプラクティス に対してポリシーチェックを行います。IAM Access Analyzer を使用したポリシーの検証の詳細については、IAM ユーザーガイドの IAM Access Analyzer のポリシーの検証を参照してください。	2021 年 11 月 30 日
新しいイベントタイプ	Amazon S3 イベント通知に追加されたイベントタイプ。Amazon S3 イベント通知を参照してください。	2021 年 11 月 29 日
バケットで Amazon EventBridge を有効にする	Amazon S3 バケットで EventBridge を有効にして Amazon EventBridge にイベントを送信できます。EventBridge を使用する を参照してください。	2021 年 11 月 29 日

新しい S3 ライフサイクルフィルタ	オブジェクトサイズに基づいてライフサイクルルールを作成したり、最新ではないオブジェクトバージョンをどれほど保持するかを指定したりできます。ライフサイクル設定についての詳細は、「S3 ライフサイクル設定の例」を参照してください。	2021 年 11 月 23 日
Amazon S3 ストレージレンズのメトリクスを Amazon CloudWatch にパブリッシュする	S3 Storage Lens の使用状況とアクティビティのメトリクスを Amazon CloudWatch に公開して、CloudWatch ダッシュボード で運用状態の統一ビューを作成できます。また、アラームやトリガーアクション、メトリクス計算、異常検出などの CloudWatch 機能を使用して、S3 Storage Lens メトリクスをモニタリングして対処することができます。さらに、 CloudWatch API により、サードパーティープロバイダーを含むアプリケーションが S3 Storage Lens メトリクスにアクセスできるようになります。詳細については、Cloudwatch の S3 Storage Lens メトリクスのモニタリング を参照してください。	2021 年 11 月 22 日

マルチリージョンアクセスポイント	マルチリージョンアクセスポイントを使用して、アプリケーションが複数の AWS リージョン にある Amazon S3 バケットからのリクエストを実行するために使用できるグローバルエンドポイントを作成できます。このマルチリージョンアクセスポイントを使用して、レイテンシーが最も低いバケットにデータをルーティングできます。マルチリージョンアクセスポイントの詳細と使用方法については、「Amazon S3 のマルチリージョンアクセスポイント」を参照してください。	2021 年 9 月 2 日

Amazon S3 on Outposts に追加されたアプリケーションのダイレクトローカルアクセス	AWS Outposts 仮想プライベートクラウド (VPC) の外でアプリケーションを実行し、S3 on Outposts のデータにアクセスすることができます。オンプレミスネットワークから直接 S3 on Outposts オブジェクトにアクセスすることもできます。お客様所有の IP (CoIP) アドレスを使用して S3 on Outposts のエンドポイントを構成し、オンプレミスネットワークからローカルゲートウェイを作成してオブジェクトにアクセスする方法については、VPC 専用のアクセスポイントを使用した Amazon S3 on Outposts へのアクセスに関する記事を参照してください。	2021 年 7 月 29 日
Amazon S3 アクセスポイントエイリアス	アクセスポイントを作成すると、Amazon S3 はデータアクセス用のバケット名の代わりに使用できるエイリアスを自動的に生成します。このアクセスポイントエイリアスは、あらゆるアクセスポイントのデータプレーンオペレーションにおいて、Amazon リソースネーム (ARN) の代わりに使用することができます。詳細については、「アクセスポイントにバケット形式のエイリアスを使用する」を参照してください。	2021 年 7 月 26 日

Amazon S3 インベントリと S3 バッチオペレーションでは、S3 バケットキーのステータスがサポートされています	Amazon S3 インベントリおよびバッチオペレーションでは、S3 バケットキーを使用した既存のオブジェクトの識別とコピーがサポートされています。S3 バケットキーは、既存のオブジェクトのサーバー側の暗号化コストの削減を加速します。詳細については、「Amazon S3 Inventory」および「バッチオペレーションによるオブジェクトのコピー」を参照してください。	2021 年 6 月 3 日
Amazon S3 ストレージレンズメトリクスアカウントのスナップショット	S3 Storage Lens アカウントのスナップショットは、デフォルトのダッシュボードからメトリクスを要約して、S3 コンソールホーム ([Buckets] (バケット)) ページに、ストレージ合計、オブジェクト数、および平均オブジェクトサイズを表示します。詳細については、S3 Storage Lens メトリクスアカウントのスナップショットを参照してください。	2021 年 5 月 5 日
強化された Amazon S3 on Outposts のエンドポイントサポート	S3 on Outposts では、Outposts ごとに最大 100 個のエンドポイントがサポートされるようになりました。詳細については、S3 on Outposts のネットワーク制限を参照してください。	2021 年 4 月 29 日

Amazon CloudWatch Events での Amazon S3 on Outposts イベント通知	CloudWatch Events を使用して、S3 on Outposts の API イベントをキャプチャし、サポートされているすべての CloudWatch ターゲットを通じて通知を受け取るためのルールを作成できます。詳細については、CloudWatch Events を使用した S3 on Outposts イベント通知の受信を参照してください。	2021 年 4 月 19 日
S3 Object Lambda	S3 Object Lambda を使用すると、Amazon S3 GET リクエストに独自のコードを追加して、データがアプリケーションに返されるときにそのデータを変更および処理できます。カスタムコードを使用すると、標準 S3 GET リクエストによって返されるデータを変更し、行のフィルタリング、画像の動的なサイズ変更、機密データの編集などを行うことができます。詳細については、「オブジェクトの変換」を参照してください。	2021 年 3 月 18 日

AWS PrivateLink	AWS PrivateLink for Amazon S3 を使用すると、インターネット経由で接続する代わりに、仮想プライベートクラウド (VPC) のインターフェイスエンドポイントを使用して S3 に直接接続できます。インターフェイスエンドポイントは、オンプレミスまたは別の AWS リージョン にあるアプリケーションから直接アクセスできます。詳細については、「AWS PrivateLinkAmazon S3」を参照してください。	2021 年 2 月 2 日
次を使用した Amazon S3 on Outposts キャパシティーの管理AWS CloudTrail	S3 on Outposts 管理イベントは、CloudTrail ログを通してご利用いただけます。詳細については、CloudTrail を使用した S3 on Outposts キャパシティーの管理を参照してください。	2020 年 12 月 21 日

強力な整合性	Amazon S3 には、すべての AWS リージョン にある Amazon S3 バケットの、オブジェクトの PUT および DELETE リクエストに関する、書き込み後読み取りの強力な整合性があります。さらに、Amazon S3 Select、Amazon S3 アクセスコントロールリスト、Amazon S3 オブジェクトタグ、オブジェクトメタデータ (HEAD オブジェクトなど) での読み込みオペレーションには強力な整合性があります。詳細については、「Amazon S3 のデータ整合性モデル」を参照してください。	2020 年 12 月 1 日
Amazon S3 レプリカ変更の同期	Amazon S3 レプリカ変更を同期することにより、ソースオブジェクトとレプリカとを同期する際に、タグ、ACL、オブジェクトロック設定などのオブジェクトメタデータを維持できます。この機能を有効にすると、Amazon S3 は、ソースオブジェクトまたはレプリカコピーのいずれかに対して行われたメタデータの変更を、レプリケートします。詳細については、「レプリカ変更の同期を使用してメタデータの変更をレプリケートする」を参照してください。	2020 年 12 月 1 日

Amazon S3 バケットキー	Amazon S3 バケットキーは、AWS Key Management Service (SSE-KMS) を使用した Amazon S3 サーバー側の暗号化のコストを削減します。サーバー側の暗号化でこの新しいバケットレベルキーを使用すると、Amazon S3 から AWS KMS へのリクエストトラフィックを減らすことにより、AWS KMS リクエストコストを最大 99% 削減できます。詳細については、「S3 バケットキーを使用した SSE-KMS のコストの削減」を参照してください。	2020 年 12 月 1 日

Amazon S3 Storage Lens	S3 ストレージレンズはメトリクスを集約し、Amazon S3 コンソールの [Buckets] (バケット) ページの [Account snapshot] (アカウントスナップショット) セクションにこの情報を表示します。S3 Storage Lens は、インサイトと傾向を可視化したり、外れ値にフラグ付けしたり、ストレージコストの最適化やデータ保護のベストプラクティスの適用に関するレコメンデーション事項を受け取ったりするために使用できるインタラクティブダッシュボードも提供します。ダッシュボードには、組織、アカウント、AWS リージョン、バケット、オブジェクト、またはプレフィックス、または Storage Lens グループレベルでインサイトを生成して可視化できる、ドリルダウンオプションが用意されています。1 日 1 回のメトリクスのエクスポートを CSV 形式または Parquet 形式で S3 バケットに送信することもできます。詳細については、「S3 Storage Lens を使用したストレージのアクティビティと使用状況の評価」を参照してください。	2020 年 11 月 18 日

次を使用した S3 リクエストのトレースAWS X-Ray	Amazon S3 は X-Ray と統合して、トレースコンテキストを伝達し、アップストリームノードとダウンストリームノードを持つ 1 つのリクエストチェーンを提供します。詳細については、「X-Ray を使用したリクエストのトレース」を参照してください。	2020 年 11 月 16 日
S3 レプリケーションメトリクス	S3 レプリケーションメトリクスは、レプリケーション設定のレプリケーションルールの詳細なメトリクスを提供します。詳細については、「レプリケーションメトリクスと Amazon S3 イベント通知」を参照してください。	2020 年 11 月 9 日
S3 Intelligent-Tiering のアーカイブアクセスとディープアーカイブアクセス	S3 Intelligent-Tiering のアーカイブアクセスとディープアーカイブアクセスは、S3 Intelligent-Tiering の下位にあるストレージ層です。詳細については、「アクセスが頻繁なオブジェクトと頻繁ではないオブジェクトを自動的に最適化するストレージクラス」を参照してください。	2020 年 11 月 9 日

削除マーカーレプリケーション	削除マーカーのレプリケーションを使用すると、レプリケーションルールの送信先のバケットに、削除マーカーを確実にコピーできます。詳細については、「削除マーカーレプリケーションの使用」を参照してください。	2020 年 11 月 9 日
S3 オブジェクトの所有権	オブジェクト所有権は、バケットにアップロードされる新しいオブジェクトの所有権を制御するために使用できる S3 バケット設定です。詳細については、「S3 オブジェクト所有権の使用」を参照してください。	2020 年 10 月 2 日
Amazon S3 on Outposts	Amazon S3 on Outposts を使用すると、AWS Outposts リソースで S3 バケットを作成し、ローカルデータアクセス、ローカルデータ処理、データレジデンシーを必要とするアプリケーション用に、オンプレミスのオブジェクトを簡単に保存および取得できます。AWS Management Console、AWS CLI、AWS SDK、または REST API を使用して S3 on Outposts を使用できます。詳細については、Amazon S3 on Outposts の使用を参照してください。	2020 年 9 月 30 日

バケット所有者条件	Amazon S3 バケット所有者条件を使用すると、S3 オペレーションで使用するバケットが、正しい AWS アカウント に属していることを確認できます。詳細については、「バケット所有者条件」を参照してください。	2020 年 9 月 11 日
オブジェクトロック保持のための S3 バッチオペレーションサポート	バッチ操作と S3 オブジェクトロックを使用して、多くの Amazon S3 オブジェクトに一括で保持設定を適用できます。詳細については、「Setting S3 Object Lock Retention dates with S3 Batch Operations」(S3 バッチオペレーションを使用した S3 オブジェクトロック保持日の設定) を参照してください。	2020 年 5 月 4 日
オブジェクトロックのリーガルホールドのための S3 バッチオペレーションサポート	バッチオペレーションと S3 オブジェクトロックを使用して、多数の Amazon S3 オブジェクトに一括でリーガルホールドを追加できるようになりました。詳細については、「S3 オブジェクトロックの法的保留の設定への S3 バッチオペレーションの使用」を参照してください。	2020 年 5 月 4 日

S3 バッチオペレーションのジョブタグ	S3 バッチオペレーションジョブにタグを追加して、それらのジョブを制御およびラベル付けできます。詳細については、「Tags for S3 Batch Operations jobs」(S3 バッチオペレーションジョブのタグ) を参照してください。	2020 年 3 月 16 日
Amazon S3 アクセスポイント	Amazon S3 アクセスポイントは、S3 の共有データセットへの大規模なデータアクセスの管理を簡素化します。アクセスポイントは、バケットにアタッチされた名前付きのネットワークエンドポイントで、S3 オブジェクトのオペレーションを実行するために使用できます。詳細については、Amazon S3 アクセスポイントによるデータアクセスの管理を参照してください。	2019 年 12 月 2 日
Access Analyzer for Amazon S3	Access Analyzer for Amazon S3 は、インターネットの任意のユーザーや他の AWS アカウント (組織外のアカウントを含む) にアクセスを許可するように設定されている S3 バケットに関して警告します。詳細については、「Access Analyzer for Amazon S3 の使用」を参照してください。	2019 年 12 月 2 日

S3 Replication Time Control (S3 RTC)	S3 Replication Time Control (S3 RTC) では、Amazon S3 に「アップロードする」ほとんどのオブジェクトを数秒で、また 99.99% のオブジェクトを 15 分以内でレプリケートします。詳細については、「S3 Replication Time Control (S3 RTC) によるオブジェクトのレプリケーション」を参照してください。	2019 年 11 月 20 日
同一リージョンレプリケーション	同じ AWS リージョン 内の Amazon S3 バケット間でオブジェクトをコピーするには、同一リージョンレプリケーション (SRR) を使用します。クロスリージョンレプリケーション (CRR) および同一リージョンレプリケーションの詳細については、「レプリケーション」を参照してください。	2019 年 9 月 18 日
S3 オブジェクトロックのクロスリージョンレプリケーションのサポート	クロスリージョンレプリケーションはオブジェクトロックをサポートするようになりました。詳細については、「Amazon S3 がレプリケートするもの」を参照してください。	2019 年 5 月 28 日

S3 バッチオペレーション	S3 バッチオペレーションを使用することで、Amazon S3 のオブジェクトに対して大規模なバッチオペレーションを実行できます。S3 バッチオペレーションでは、指定したオブジェクトのリストに対して、1 つのオペレーションを実行できます。1 つのジョブで、エクサバイトのデータを含む数十億ものオブジェクトに対して、指定されたオペレーションを実行できます。詳細については、「S3 バッチオペレーションの実行」を参照してください。	2019 年 4 月 30 日
アジアパシフィック (香港) リージョン	Amazon S3 が、アジアパシフィック (香港) リージョンで使用できるようになりました。Amazon S3 リージョンおよびエンドポイントの詳細については、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。	2019 年 4 月 24 日
サーバーアクセスログに新しいフィールドを追加しました	Amazon S3 のサーバーアクセスログに次の新しいフィールドが追加されました: TLS (Transport Layer Security (TLS)) バージョン。詳細については、「サーバーアクセスログの形式」を参照してください。	2019 年 3 月 28 日

新しいアーカイブストレージクラス	Amazon S3 で、アクセス頻度の低いオブジェクトの保存用に、新しいアーカイブストレージクラス「S3 Glacier Deep Archive (DEEP_ARCHIVE)」を利用できるようになりました。詳細については、「ストレージクラス」を参照してください。	2019 年 3 月 27 日
新しいフィールドをサーバーアクセスログに追加しました	Amazon S3 では、次の新しいフィールドをサーバーアクセスログに追加しました: ホスト ID、署名バージョン、暗号スイート、認証タイプ、ホストヘッダー 詳細については、「サーバーアクセスログの形式」を参照してください。	2019 年 3 月 5 日
Parquet 形式の Amazon S3 インベントリファイルのサポート	Simple Storage Service (Amazon S3) では、Apache optimized row columnar (ORC)およびインベントリ出力ファイルのカンマ区切り値 (CSV) ファイル形式に加えて、Apache Parquet (Parquet) 形式をサポートします。詳細については、「インベントリ」を参照してください。	2018 年 12 月 4 日

S3 オブジェクトロック	Amazon S3 で、Amazon S3 オブジェクトへの Write Once Read Many (WORM) 保護を提供するオブジェクトロック機能を利用できるようになりました。詳細については、「オブジェクトのロック」を参照してください。	2018 年 11 月 26 日
復元速度のアップグレード	Amazon S3 の復元速度のアップグレードを使用すると、S3 Glacier Flexible Retrieval ストレージクラスで復元速度を変更できるため、復元が進行中であっても速度をより速くできます。詳細については、「アーカイブされたオブジェクトの復元」を参照してください。	2018 年 11 月 26 日
イベント通知の復元	Amazon S3 イベント通知では、オブジェクトを S3 Glacier Flexible Retrieval ストレージクラスから復元する際の開始イベントと完了イベントがサポートされるようになりました。詳細については、「イベント通知」を参照してください。	2018 年 11 月 26 日

S3 Glacier Flexible Retrieval ストレージクラスへの PUT ディレクトリ	Amazon S3 PUT オペレーションで、オブジェクトを作成する際に S3 Glacier Flexible Retrieval をストレージクラスとして指定できるようになりました。以前は、オブジェクトを別の Amazon S3 ストレージクラスから S3 Glacier Flexible Retrieval ストレージクラスに移行する必要がありました。さらに S3 クロスリージョンレプリケーション (CRR) を使用することで、S3 Glacier Flexible Retrieval をレプリケートされたオブジェクトのストレージクラスとして指定することもできるようになりました。S3 Glacier Flexible Retrieval ストレージクラスの詳細については、「ストレージクラス」を参照してください。レプリケートされたオブジェクトのストレージクラスを指定する方法の詳細については、「レプリケーション設定の概要」を参照してください。S3 Glacier Flexible Retrieval REST API 変更への PUT ディレクトリについての詳細は、ドキュメント履歴: S3 Glacier Flexible Retrieval への PUT ディレクトリを参照してください。	2018 年 11 月 26 日

新しいストレージクラス	Amazon S3 では、S3 Intelligent-Tiering (`INTELLIGENT_TIERING`) という名前のストレージクラスの提供を開始しました。このストレージクラスは、アクセスパターンが変化する、または未知で、長期間使用するデータ用に設計されています。詳細については、「ストレージクラス」を参照してください。	2018 年 11 月 26 日
Amazon S3 パブリックアクセスブロック	Amazon S3 にバケット単位またはアカウント全体でバケットとオブジェクトへのパブリックアクセスをブロックする機能が追加されました。詳細については、「Amazon S3 パブリックアクセスのブロックの使用」を参照してください。	2018 年 11 月 15 日

クロスリージョンレプリケーション (CRR) ルールのフィルタ処理機能の強化	CRR ルール設定では、オブジェクトフィルタを指定して、ルールを適用するオブジェクトのサブセットを選択できます。これまでは、オブジェクトキープレフィックスでのみフィルタを適用できました。今回のリリースにより、オブジェクトキープレフィックス、1 つ以上のオブジェクトタグ、またはその両方でフィルタを適用できるようになりました。詳細については、「CRR セットアップ: レプリケーション設定の概要」を参照してください。	2018 年 9 月 19 日
Amazon S3 Select の新機能	Amazon S3 Select では、Apache Parquet 入力、ネストされた JSON オブジェクトでのクエリ、および 2 つの新しい Amazon CloudWatch モニタリングメトリクス (`SelectScannedBytes` と `SelectReturnedBytes`) がサポートされるようになりました。	2018 年 9 月 5 日
更新を RSS で今すぐ入手可能	RSS フィードにサブスクライブすると、Amazon S3 ユーザーガイドの更新に関する通知を受け取ることができます。	2018 年 6 月 19 日

以前の更新

次の表に、2018 年 6 月 19 日以前の Amazon S3 ユーザーガイドの各リリースにおける重要な変更点が記されています。

変更	説明	日付
コード例の更新	コード例の更新: • C#—タスクベースの非同期パターンを使用するようにすべての例を更新しました。詳細については、AWS SDK for .NET デベロッパーガイドの「Amazon Web Services Asynchronous APIs for .NET」(.NET の Amazon Web Services 非同期 API) を参照してください。コード例は、バージョン 3 の AWS SDK for .NET に準拠するようになりました。 • Java—クライアントビルダーモデルを使用するようにすべての例を更新しました。クライアントビルダーモデルの詳細については、「サービスクライアントの作成」を参照してください。 • PHP – AWS SDK for PHP 3.0 を使用するようにすべての例を更新しました。AWS SDK for PHP 3.0 の詳細については、「AWS SDK for PHP」を参照してください。 • Ruby – サンプルコードを更新し、例が AWS SDK for Ruby バージョン 3 で動作するようにしました。	2018 年 30 月 4 日
Amazon S3 は、S3 Glacier Flexible Retrieval および `ONEZONE_IA` ストレージクラスを Amazon CloudWatch Logs ストレージメト	これらのストレージメトリクスには、実際のバイト数の報告に加えて、該当するストレージクラス (`ONEZONE_IA` 、`STANDARD_IA` 、S3 Glacier Flexible Retrieval) のオブジェクトごとのオーバーヘッドバイト数も含まれます。 • `ONEZONE_IA` および `STANDARD_IA` ストレージクラスオブジェクトの場合、Amazon S3 は 128 KB より小さいオブジェクトを 128 KB と報告します。詳細について	2018 年 30 月 4 日

変更	説明	日付
リクスに報告するようになりました。	は、「Amazon S3 ストレージクラスを使用する」を参照してください。 • S3 Glacier Flexible Retrieval ストレージクラスオブジェクトの場合、ストレージメトリクスは次のオーバーヘッドを報告します。 • オブジェクトあたり 32 KB のオーバーヘッド、S3 Glacier Flexible Retrieval ストレージクラスの料金で請求 • オブジェクトあたり 8 KB のオーバーヘッド、STANDARD ストレージクラスの料金で請求 詳細については、「Amazon S3 ライフサイクルを使用したオブジェクトの移行」を参照してください。 ストレージメトリクスの詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。	
新しいストレージクラス	Amazon S3 では、オブジェクトを保存するための新しいストレージクラスである STANDARD_IA (IA は少頻度アクセスを意味します) の提供が開始されました。このストレージクラスは、存続時間が長く、アクセス頻度の低いデータに最適化されています。詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。	2018 年 4 月 4 日
Amazon S3 Select	Amazon S3 は、SQL 式に基づいたオブジェクトのコンテンツの取得をサポートするようになりました。詳細については、「Amazon S3 Select を使用したデータのフィルタリングと取得」を参照してください。	2018 年 4 月 4 日

変更	説明	日付
アジアパシフィック (大阪: ローカル) リージョン	Amazon S3 が、アジアパシフィック (大阪ローカル) リージョンで使用できるようになりました。Amazon S3 リージョンおよびエンドポイントの詳細については、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。 **Important** アジアパシフィック (大阪ローカル) リージョンは、アジアパシフィック (東京) リージョンと組み合わせてのみ使用できます。アジアパシフィック (大阪ローカル) リージョンへのアクセスをリクエストするには、販売担当者にお問い合わせください。	2018 年 2 月 12 日
Amazon S3 インベントリ作成のタイムスタンプ	Amazon S3 インベントリには、Amazon S3 インベントリレポートの作成日と作成開始時刻のタイムスタンプが含まれるようになりました。このタイムスタンプを使用して、インベントリレポートの作成開始時刻以降の Amazon S3 ストレージの変更を判断できます。	2018 年 1 月 16 日
欧州 (パリ) リージョン	Amazon S3 が欧州 (パリ) リージョンで利用可能になりました。Amazon S3 リージョンおよびエンドポイントの詳細については、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。	2017 年 18 月 12 日
中国 (寧夏) リージョン	Amazon S3 が中国 (寧夏) リージョンで利用可能になりました。Amazon S3 リージョンおよびエンドポイントの詳細については、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。	2017 年 11 月 29 日

変更	説明	日付
ORC 形式の Amazon S3 インベントリファイルのサポート	Amazon S3 は、インベントリ出力ファイルにカンマ区切り値 (CSV) ファイル形式に加えて Apache 最適化行列 (ORC) 形式をサポートするようになりました。また、Amazon S3 インベントリをクエリする際に、Amazon Athena、Amazon Redshift Spectrum、およびその他のツール (Presto、Apache Hive、Apache Spark など) で標準の SQL を使用できるようになりました。詳細については、「Amazon S3 インベントリ」を参照してください。	2017 年 11 月 17 日
S3 バケットのデフォルトの暗号化	Amazon S3 のデフォルトの暗号化により、S3 バケットのデフォルト暗号化の動作を設定できます。バケットにデフォルト暗号化を設定して、バケットに保存される際すべてのオブジェクトが暗号化されるようにします。オブジェクトは、Amazon S3 マネージドキー (SSE-S3) と AWS マネージドキー (SSE-KMS) のいずれかで、サーバー側の暗号化を使って暗号化されます。詳細については、「Amazon S3 バケット向けのサーバー側のデフォルトの暗号化動作の設定」を参照してください。	2017 年 11 月 06 日
Amazon S3 インベントリの暗号化ステータス	Amazon S3 が Amazon S3 インベントリでの暗号化ステータスをサポートするようになったため、オブジェクトが保管時にどのようにコンプライアンス監査などの目的で暗号化されているかを確認できます。また、Amazon S3 インベントリをサーバー側暗号化 (SSE) または SSE-KMS で暗号化するように設定でき、すべてのインベントリファイルはそれに応じて暗号化されます。詳細については、「Amazon S3 インベントリ」を参照してください。	2017 年 11 月 06 日

変更	説明	日付
クロスリージョンレプリケーション (CRR) の機能強化	クロスリージョンレプリケーションは以下をサポートするようになりました。 • クロスアカウントのシナリオでは、CRR の設定を追加してレプリカの所有権を、送信先バケットを所有する AWS アカウント に変更できます。詳細については、「レプリカ所有者の変更」を参照してください。 • デフォルトでは、Amazon S3 は AWS KMS に保存されたキーでサーバー側の暗号化を使用して作成されたソースバケット内のオブジェクトをレプリケートしません。CRR 設定で、これらのオブジェクトをレプリケートするよう Amazon S3 に指示できるようになりました。詳細については、「暗号化されたオブジェクトのレプリケート (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)」を参照してください。	2017 年 11 月 06 日
欧州 (ロンドン) リージョン	Amazon S3 が欧州 (ロンドン) リージョンで利用可能になりました。Amazon S3 リージョンおよびエンドポイントの詳細については、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。	2016 年 12 月 13 日
カナダ (中部) リージョン	Amazon S3 がカナダ (中部) リージョンで使用可能になりました。Amazon S3 リージョンおよびエンドポイントの詳細については、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。	2016 年 12 月 8 日

変更	説明	日付
オブジェクトのタグ付け	Amazon S3 では、オブジェクトのタグ付けがサポートされるようになりました。オブジェクトのタグ付けによりストレージを分類できます。オブジェクトキー名のプレフィックスによってもストレージを分類でき、オブジェクトのタグ付けによって別のディメンションが追加されます。 タグ付けには他のメリットもあります。具体的には次のとおりです。 • オブジェクトタグにより、アクセス許可を細かくアクセスコントロールできます (たとえば、特定のタグが付いたオブジェクトのみ読み取るアクセス許可を IAM ユーザーに付与できます)。 • ライフサイクル設定を指定する際の細かいコントロール。ライフサイクルルールが適用されるオブジェクトのサブセットを選択するタグを指定できます。 • クロスリージョンレプリケーション (CRR) を設定している場合、Amazon S3 はタグをレプリケートできます。オブジェクトを自動的にレプリケートできるように、Amazon S3 に作成された IAM ロールに必要なアクセス許可を付与する必要があります。 • CloudWatch メトリクスと CloudTrail イベントをカスタマイズして、特定のタグフィルタごとに情報を表示することもできます。 詳細については、「タグを使用してストレージを分類する」を参照してください。	2016 年 11 月 29 日

変更	説明	日付
Amazon S3 ライフサイクルでタグベースのフィルタのサポートを開始	Amazon S3 では、ライフサイクル設定におけるタグベースのフィルタリングがサポートされるようになりました。キープレフィックス、1 つ以上のオブジェクトタグ、または両方の組み合わせを指定できるライフサイクルルールを指定して、ライフサイクルルールが適用されるオブジェクトのサブセットを選択できるようになりました。詳細については、「ストレージのライフサイクルの管理」を参照してください。	2016 年 11 月 29 日
バケットの CloudWatch リクエストメトリクス	Amazon S3 では、バケットで行われたリクエストの CloudWatch メトリクスがサポートされるようになりました。バケットでこれらのメトリクスを有効にすると、メトリクスが 1 分ごとに報告されます。これらのリクエストメトリクスを報告するバケット内のオブジェクトを設定することもできます。詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。	2016 年 11 月 29 日
Amazon S3 インベントリ	Amazon S3 では、ストレージインベントリがサポートされるようになりました。Amazon S3 インベントリは、S3 バケットまたは共有プレフィックスに関して、オブジェクトのフラットファイル出力と対応するメタデータを毎日または毎週生成します (つまり、名前の先頭が共通文字列のオブジェクト)。 詳細については、「Amazon S3 インベントリ」を参照してください。	2016 年 11 月 29 日

変更	説明	日付
Amazon S3 分析 – ストレージクラス分析	新しい Amazon S3 分析 – ストレージクラスの分析機能は、アクセス頻度の低い STANDARD ストレージをいつ STANDARD_IA (IA: 小頻度アクセス) ストレージクラスに移行すべきかを判断できるように、データアクセスパターンを確認します。ストレージクラス分析が、フィルタリングされたデータセットの小頻度アクセスパターンを一定期間監視すると、分析結果を使用してライフサイクル設定を改善できます。この機能には、S3 バケットにエクスポートできる指定したバケット、プレフィックス、またはタグのレベルに応じたストレージ使用状況の毎日の詳細な分析も含まれています。	2016 年 11 月 29 日
アーカイブされたオブジェクトを S3 Glacier から復元する際の新しい緊急データ取得と一括データ取得	Amazon S3 では、S3 Glacier にアーカイブされたオブジェクトを復元する際に、標準取得に加えて緊急データ取得と一括データ取得がサポートされるようになりました。詳細については、「アーカイブされたオブジェクトの復元」を参照してください。	2016 年 11 月 21 日
CloudTrail オブジェクト記録	CloudTrail では、GetObject 、PutObject 、DeleteObject など、Amazon S3 オブジェクトレベルの API オペレーションのログ記録がサポートされます。オブジェクトレベルの API オペレーションが口グ記録されるように、イベントセレクターを設定できます。詳細については、「AWS CloudTrail を使用した Amazon S3 API コールのログ記録」を参照してください。	2016 年 11 月 21 日
米国東部 (オハイオ) リージョン	Amazon S3 が米国東部 (オハイオ) リージョンで利用可能になりました。Amazon S3 リージョンおよびエンドポイントの詳細については、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。	2016 年 10 月 17 日

変更	説明	日付
Amazon S3 Transfer Acceleration 向けの IPv6 サポート	Amazon S3 は、Amazon S3 Transfer Acceleration 向けのインターネットプロトコルバージョン 6 (IPv6) をサポートするようになりました。Transfer Acceleration エンドポイントの新しいデュアルスタックを使用することで、IPv6 を使って Amazon S3 に接続できます。詳細については、「Amazon S3 Transfer Acceleration の開始方法」を参照してください。	2016 年 10 月 6 日
IPv6 サポート	Amazon S3 がインターネットプロトコルのバージョン 6 (IPv6) をサポートするようになりました。デュアルスタックのエンドポイントを使用して IPv6; で Amazon S3 にアクセスできます。詳細については、「IPv6 経由で Amazon S3 へのリクエストを行う」を参照してください。	2016 年 8 月 11 日
アジアパシフィック (ムンバイ) リージョン	Amazon S3 が、アジアパシフィック (ムンバイ) リージョンで使用できるようになりました。Amazon S3 リージョンおよびエンドポイントの詳細については、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。	2016 年 6 月 27 日
Amazon S3 Transfer Acceleration	Amazon S3 Transfer Acceleration によって、クライアントと S3 バケットの長距離間でファイルを高速、簡単、安全に転送できます。Transfer Acceleration は、Amazon CloudFront の世界中に点在するエッジロケーションを利用します。 詳細については、「Amazon S3 Transfer Acceleration を使用した高速かつ安全なファイル転送の設定」を参照してください。	2016 年 4 月 19 日
期限切れオブジェクト削除マーカーを削除するライフサイクルのサポート	ライフサイクル設定の `Expiration` アクションで、バージョニング管理されたバケット内の期限切れオブジェクトの削除マーカーを削除するように Amazon S3 に指示できるようになりました。詳細については、「ライフサイクルアクションを記述する要素」を参照してください。	2016 年 3 月 16 日

変更	説明	日付
バケットのライフサイクル設定での完了していないマルチパートアップロードを中止するアクションのサポート	バケットのライフサイクル設定で、AbortIncompleteMultipartUpload アクションがサポートされるようになりました。このアクションを使用すると、開始後に指定された日数以内に完了していないマルチパートアップロードを中止するように Amazon S3 に指示できます。マルチパートアップロードが中止オペレーションの対象になると、Amazon S3 はアップロードされたパートをすべて削除し、マルチパートアップロードを中止します。 概念的な情報については、Amazon S3 ユーザーガイドの以下のトピックを参照してください。 • マルチパートアップロードの中止 • ライフサイクルアクションを記述する要素 次の API オペレーションが更新され、新しいアクションがサポートされるようになりました。 • PUT Bucket lifecycle – XML 設定で、ライクサイクル設定ルールに AbortIncompleteMultipartUpload アクションを指定できるようになりました。 • パートの一覧表示とマルチパートアップロードの開始 – バケットに x-amz-abort-date アクションを指定しているライフサイクルルールがある場合、これらの 2 つの API オペレーションは追加の 2 つのレスポンスヘッダー (x-amz-abort-rule-id と AbortIncompleteMultipartUpload) を返すようになりました。レスポインスのこれらのヘッダーは、開始されたマルチパートアップロードがいつ中止オペレーションの対象になり、どのライフサイクルルールが適用されるかを示します。	2016 年 3 月 16 日

変更	説明	日付
アジアパシフィック (ソウル) リージョン	Amazon S3 が、アジアパシフィック (ソウル) リージョンで使用できるようになりました。Amazon S3 リージョンおよびエンドポイントの詳細については、「AWS 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。	2016 年 1 月 6 日
新しい条件キーとマルチパートアップロードの変更	IAM ポリシーが、Amazon S3 `s3:x-amz-storage-class` 条件キーをサポートするようになりました。詳細については、「条件キーを使用したバケットポリシーの例」を参照してください。 パートをアップロードしてアップロードを完了するために、マルチパートアップロードのイニシエータになる必要がなくなりました。詳細については、「マルチパートアップロード API とアクセス許可」を参照してください。	2015 年 12 月 14 日
米国スタンダードリージョンの名前を変更しました	「米国スタンダード」から「米国東部 (バージニア北部)」にリージョン名文字列を変更しました。これはリージョン名の更新のみであり、機能に変更はありません。	2015 年 12 月 11 日

変更	説明	日付
新しいストレージクラス	Amazon S3 では、オブジェクトを保存するための新しいストレージクラスである STANDARD_IA (IA、少頻度アクセス) の提供が開始されました。このストレージクラスは、存続時間が長く、アクセス頻度の低いデータに最適化されています。詳細については、「Amazon S3 ストレージクラスを使用する」を参照してください。 ライフサイクル設定機能の更新により、STANDARD_IA ストレージクラスにオブジェクトを移行できるようになりました。詳細については、「ストレージのライフサイクルの管理」を参照してください。 これまでは、クロスリージョンレプリケーション機能では、オブジェクトのレプリカにソースオブジェクトのストレージクラスが使用されました。現在は、クロスリージョンのレプリケーションを設定するときに、送信先バケットで作成されたオブジェクトレプリカ用にストレージクラスを指定することができます。詳細については、「オブジェクトのレプリケーション」を参照してください。	2015 年 9 月 16 日
AWS CloudTrail の統合	新しい AWS CloudTrail の統合により、Amazon S3 API 操作を S3 バケットに記録できるようになりました。CloudTrail を使用して、S3 バケットの作成または削除、アクセス制御の変更、またはライフサイクル設定の変更を追跡できます。詳細については、「AWS CloudTrail を使用した Amazon S3 API コールのログ記録」を参照してください。	2015 年 9 月 1 日

変更	説明	日付
バケット制限の引き上げ	Amazon S3 でバケット制限の引き上げがサポートされるようになりました。デフォルトでは、AWS アカウント で作成できるバケットの数は最大で 100 個です。追加のバケットを必要とする場合は、サービスの制限の引き上げを申請することで制限を引き上げることができます。バケットの制限を引き上げる方法については、AWS 全般リファレンスの「AWS のサービス クォータ」を参照してください。詳細については、AWS SDK の使用およびバケットの制約と制限を参照してください。	2015 年 8 月 4 日
整合性モデルの更新	Amazon S3 が、米国東部 (バージニア北部) リージョンで Amazon S3 に追加された新しいオブジェクトの read-after-write 整合性をサポートするようになりました。この更新の前は、米国東部 (バージニア北部) 以外のすべてのリージョンが、Amazon S3 にアップロードされた新しいオブジェクトの「read-after-write」整合性をサポートしていました。この機能強化により、Amazon S3 は、すべてのリージョンについて、Amazon S3 に追加された新しいオブジェクトの「read-after-write」整合性をサポートするようになりました。「Read-after-write」整合性により、Amazon S3 でオブジェクトを作成した直後にオブジェクトを取得できます。詳細については、「リージョン」を参照してください。	2015 年 8 月 4 日
イベント通知	Amazon S3 のイベント通知が更新されており、オブジェクトが削除されたときの通知が追加され、プレフィックスおよびサフィックスの一致によるオブジェクト名のフィルタリングが追加されました。詳細については、「Amazon S3 イベント通知」を参照してください。	2015 年 7 月 28 日

変更	説明	日付
Amazon CloudWatch との統合	新しい Amazon CloudWatch 統合により、Amazon S3 の CloudWatch メトリクスを使用して、Amazon S3 の使用状況に関するアラームを監視および設定できます。サポートされているメトリクスには、標準ストレージの合計バイト数、低冗長化ストレージの合計バイト数、特定の S3 バケットのオブジェクトの合計数などがあります。詳細については、「Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。	2015 年 7 月 28 日
空ではないバケットの削除とバケットを空にするサポート	Amazon S3 は、空ではないバケットの削除とバケットを空にするサポートができるようになりました。詳細については、「バケットを空にする」を参照してください。	2015 年 7 月 16 日
Amazon VPC エンドポイントのバケットポリシー	Amazon S3 では、仮想プライベートクラウド (VPC) エンドポイントのバケットポリシーのサポートが追加されました。S3 バケットポリシーを使用して、特定の VPC エンドポイントまたは特定の VPC からバケットへのアクセスを制御できます。VPC エンドポイントは設定が簡単で信頼性が高く、ゲートウェイまたは NAT インスタンスを必要とすることなく、Amazon S3 に安全に接続できます。詳細については、「バケットポリシーを使用した VPC エンドポイントからのアクセスコントロール」を参照してください。	2015 年 4 月 29 日
イベント通知	Amazon S3 イベント通知は、AWS Lambda 関数のリソースベースの許可への切り替えをサポートするように更新されました。詳細については、「Amazon S3 イベント通知」を参照してください。	2015 年 4 月 9 日
クロスリージョンレプリケーション	Amazon S3 がクロスリージョンレプリケーションをサポートするようになりました。クロスリージョンレプリケーションは、異なる AWS リージョンにあるバケット間でオブジェクトを自動的に非同期コピーする機能です。詳細については、「オブジェクトのレプリケーション」を参照してください。	2015 年 3 月 24 日

変更	説明	日付
イベント通知	Amazon S3 では、バケットの通知設定で、新しいイベントタイプと宛先がサポートされるようになりました。このリリース以前の Amazon S3 では、s3:ReducedRedundancyLostObject イベントタイプおよびその宛先として Amazon SNS トピックのみがサポートされていました。新しいイベントタイプの詳細については、「Amazon S3 イベント通知」を参照してください。	2014 年 11 月 13 日
お客様が用意した暗号化キーを使用したサーバー側の暗号化	AWS Key Management Service (AWS KMS) キー (SSE-KMS) によるサーバー側の暗号化 Amazon S3 で、AWS KMS を使用したサーバー側暗号化がサポートされるようになりました。この機能により、AWS KMS を介してエンベロープキーを管理できます。また、Amazon S3 は AWS KMS を呼び出して、設定された許可内のエンベロープキーにアクセスできます。 AWS KMS のサーバー側の暗号化の詳細については、「AWS Key Management Service でのサーバー側の暗号化を使用したデータの保護」を参照してください。	2014 年 11 月 12 日
欧州 (フランクフルト) リージョン	Amazon S3 が欧州 (フランクフルト) リージョンで利用可能になりました。	2014 年 10 月 23 日

変更	説明	日付
お客様が用意した暗号化キーを使用したサーバー側の暗号化	Amazon S3 で、お客様が用意した暗号化キーによるサーバー側の暗号化 (SSE-C)がサポートされるようになりました。サーバー側の暗号化によって、保管時のデータの暗号化を Amazon S3 にリクエストすることができます。SSE-C を使用する場合、Amazon S3 はお客様が用意した独自の暗号化キーを使用してオブジェクトを暗号化します。Amazon S3 によって暗号化が実行されるため、独自の暗号化コードを作成または実行するコストをかけずに、独自の暗号化キーを使用できるというメリットがあります。 SSE-C の詳細については、「お客様が用意した暗号化キーによるサーバー側の暗号化 (SSE-C) を使用したデータの保護」を参照してください。	2014 年 6 月 12 日
バージョニングのライフサイクルのサポート	以前のリリースでは、ライフサイクルの設定は、バージョニングが設定されていないバケットでのみサポートされていました。このリリースでは、バージョニング対応および非対応の両方のバケットでライフサイクルを設定できます。詳細については、「ストレージのライフサイクルの管理」を参照してください。	2014 年 5 月 20 日
アクセスコントロールのトピックを改訂	Amazon S3 アクセスコントロールのドキュメントが改訂されました。詳細については、「Amazon S3 用 Identity and Access Management」を参照してください。	2014 年 4 月 15 日
サーバーアクセスの口グ記録に関するトピックが改訂されました。	改訂後の、サーバーアクセスログ記録のドキュメント。詳細については、「サーバーアクセスログによるリクエストの口グ記録」を参照してください。	2013 年 11 月 26 日
.NET SDK サンプルがバージョン 2.0 に更新	このガイドの .NET SDK サンプルがバージョン 2.0 に準拠しました。	2013 年 11 月 26 日

変更	説明	日付
HTTP 経由での SOAP のサポートが廃止される	SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。SOAP では、Amazon S3 の新機能がサポートされなくなります。REST API か AWS SDK を使用することをお勧めします。	2013 年 9 月 20 日
IAM ポリシー変数のサポート	IAM ポリシー言語では、変数がサポートされるようになりました。ポリシーが評価されると、認証ユーザーのセッションの文脈ベースの情報から得られた値によって、ポリシー変数が置換されます。ポリシー変数を使用すれば、ポリシーのすべての構成要素を明示的にリストしなくても、汎用的なポリシーを定義できます。ポリシー変数の詳細については、IAM ユーザーガイドの「IAM ポリシー変数の概要」を参照してください。 Amazon S3 のポリシー変数の例については、「Amazon S3 のアイデンティティベースのポリシー例」を参照してください。	2013 年 3 月 4 日
コンソールでのリクエスタ支払いのサポート	Amazon S3 コンソールで、バケットをリクエスタ支払い用に設定できるようになりました。詳細については、「ストレージ転送と使用量のリクエスタ支払いバケットの使用」を参照してください。	2012 年 12 月 31 日

変更	説明	日付
ウェブサイトホスティングでのルートドメインのサポート	Amazon S3 が、ルートドメインでの静的ウェブサイトのホスティングをサポートするようになりました。ウェブサイトにアクセスする訪問者がブラウザにウェブアドレスを入力する際、「www」を省略できます (例えば、www.example.com ではなく example.com を使用できます)。既に Amazon S3 上でホスティングされている静的ウェブサイトの多くが、www サブドメイン (例: www.example.com) からアクセスするようになっています。これまでは、ルートドメインアクセスをサポートするには、ルートドメインへのリクエストをブラウザから Amazon S3 上のウェブサイトに中継するためのウェブサーバーを独自に運用することが必要でした。リクエストを中継するためのウェブサーバー運用には、コストや運用の負担の増加が伴うだけではなく、障害点となりうるものが増えるおそれがあります。現在では、Amazon S3 の可用性と耐久性の高さを、「www」とルートドメインの両方のアドレスに活用できます。詳細については、「Amazon S3 を使用して静的ウェブサイトをホスティングする」を参照してください。	2012 年 12 月 27 日
コンソールの修正	Amazon S3 コンソールが更新されました。ドキュメントのトピックのうち、コンソールに言及している部分も、これに従って修正されています。	2012 年 12 月 14 日

変更	説明	日付
S3 Glacier へのデータのアーカイブのサポート	Amazon S3 で、データのアーカイブに S3 Glacier の低コストなストレージサービスを使用する新しいストレージオプションがサポートされるようになりました。オブジェクトをアーカイブするには、オブジェクト、および Amazon S3 でこれらのオブジェクトを S3 Glacier にアーカイブする期間を特定するためのアーカイブルールを定義します。Amazon S3 コンソールを使用して、あるいは Amazon S3 API または AWS SDK を使用したプログラミングによって、バケット上にルールを簡単に設定できます。 詳細については、「ストレージのライフサイクルの管理」を参照してください。	2012 年 11 月 13 日
ウェブサイトページのリダイレクトのサポート	Amazon S3 では、バケットがウェブサイトとして設定されている場合、オブジェクトに対するリクエストを同じバケット内の別のオブジェクトまたは外部 URL にリダクレクトできるようになりました。詳細については、「(オプション) ウェブページリダイレクトの設定」を参照してください。 ウェブサイトのホスティングについては、Amazon S3 を使用して静的ウェブサイトをホスティングする を参照してください。	2012 年 10 月 4 日

変更	説明	日付
Cross-Origin Resource Sharing (CORS) のサポート	Amazon S3 で Cross Origin Resource Sharing (CORS) がサポートされるようになりました。CORS は、あるドメインにロードされたクライアントウェブアプリケーションが、異なるドメイン内のリソースと通信またはアクセスする方法を定義します。Amazon S3 で CORS がサポートされたことにより、Amazon S3 を基盤とする機能豊富なクライアント側ウェブアプリケーションを構築し、Amazon S3 リソースに対するクロスドメインアクセスを選択的に許可することができます。詳細については、「Cross-Origin Resource Sharing (CORS) の使用」を参照してください。	2012 年 8 月 31 日
コスト割り当てタグのサポート	Amazon S3 でコスト割り当てタグがサポートされました。S3 バケットにラベルを付けることができるため、プロジェクトやその他の条件に照らして、それらのコストを簡単に追跡できます。バケットのタグの使用に関する詳細については、「S3 バケットタグでのコスト配分タグの使用」を参照してください。	2012 年 8 月 21 日
バケットポリシーでの MFA で保護された API アクセスのサポート	Amazon S3 で、MFA で保護された API アクセスがサポートされました。これにより、Amazon S3 リソースへのアクセス時に AWS Multi-Factor Authentication を適用してセキュリティを強化できます。このセキュリティ機能では、有効な MFA コードを入力して MFA デバイスを物理的に所有していることを証明することがユーザーに要求されます。詳細については、「AWS 多要素認証」を参照してください。Amazon S3 リソースへのすべてのアクセスリクエストに対して、MFA 認証を要求できるようになりました。 MFA 認証を適用するために、Amazon S3 がバケットポリシーで `aws:MultiFactorAuthAge` キーがサポートされるようになりました。バケットポリシーの例については、「MFA が必要」を参照してください。	2012 年 7 月 10 日

変更	説明	日付
オブジェクトの有効期限のサポート	オブジェクトの有効期限を使用して、設定した期間を経過後にデータを自動削除するよう設定できます。オブジェクトの有効期限を設定するには、バケットにライフサイクル設定を追加します。	2011 年 12 月 27 日
新しくサポートされるリージョン	Amazon S3 で、南米 (サンパウロ) リージョンがサポートされるようになりました。詳細については、「Amazon S3 バケットに対するアクセスと一覧表示」を参照してください。	2011 年 12 月 14 日
Multi-Object Delete	Amazon S3 で、単一のリクエストで複数のオブジェクトを削除できるマルチオブジェクト削除 API がサポートされるようになりました。この機能を使用して、複数の個別の DELETE リクエストを使用するよりもすばやく多数のオブジェクトを Amazon S3 から削除できます。詳細については、「Amazon S3 オブジェクトの削除」を参照してください。	2011 年 12 月 7 日
新しくサポートされるリージョン	Amazon S3 で、米国西部 (オレゴン) リージョンがサポートされるようになりました。詳細については、「バケットとリージョン」を参照してください。	2011 年 11 月 8 日
ドキュメントの更新	ドキュメントのバグの修正。	2011 年 11 月 8 日
ドキュメントの更新	ドキュメントのバグの修正に加えて、このリリースには、次の機能強化が含まれています。 • AWS SDK for PHP と AWS SDK for Ruby (「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) の指定」を参照) を使用した新しいサーバー側の暗号化セクション。	2011 年 10 月 17 日

変更	説明	日付
サーバー側暗号化のサポート	Amazon S3 で、サーバー側暗号化がサポートされるようになりました。これにより、Amazon S3 に保存時にデータを暗号化する、つまり、Amazon S3 がそのデータセンター内のディスクにオブジェクトデータを書き込むときに、データを暗号化するようにリクエストできます。REST API の更新に加えて、AWS SDK for Java および .NET には、サーバー側の暗号化をリクエストするために必要な機能が備わっています。さらに、AWS Management Console を使用して、オブジェクトのアップロード時に、サーバー側暗号化をリクエストすることもできます。データ暗号化の詳細については、「暗号化を使用したデータの保護」を参照してください。	2011 年 10 月 4 日
ドキュメントの更新	ドキュメントのバグの修正に加えて、このリリースには、次の機能強化が含まれています。 • リクエストの実行 セクションに Ruby および PHP のサンプルが追加されました。 • 署名付き URL の生成および使用方法を説明するセクションが追加されました。詳細については、署名付き URL を使用したオブジェクトの共有および署名付き URL を使用したオブジェクトの共有を参照してください。 • Eclipse および Visual Studio 用の AWS Explorer を紹介する既存のセクションが更新されました。詳細については、「AWS SDK を使用した Amazon S3 での開発」を参照してください。	2011 年 9 月 22 日

変更	説明	日付
一時的なセキュリティ認証情報を使用したリクエスト送信のサポート	AWS アカウントおよび IAM ユーザーセキュリティ証明書を使用して認証済みリクエストを Amazon S3 に送信することに加え、AWS Identity and Access Management (IAM) から取得する一時セキュリティ認証情報を使用してリクエストを送信できるようになりました。AWS Security Token Service API または AWS SDK ラッパーライブラリを使用して、一時的な認証情報を IAM でリクエストできます。これらの一時セキュリティ証明書は、自身で使用するためにリクエストするか、フェデレーティッドユーザーおよびアプリケーションに渡すことができます。この機能により、AWS の外部でユーザーを管理し、それらのユーザーに AWS リソースにアクセスするための一時セキュリティ証明書を提供できます。 詳細については、「リクエストの実行」を参照してください。 一時的なセキュリティ認証情報の IAM によるサポートの詳細については、IAM ユーザーガイドの「IAM の一時的なセキュリティ認証情報」を参照してください。	2011 年 8 月 3 日
マルチパートアップロード API 拡張による最大 5 TB のオブジェクトのコピー	このリリース以前、Amazon S3 API では5 GB までのサイズのオブジェクトのコピーをサポートしていました。5 GB 以上のオブジェクトをコピーできるようにするため、Amazon S3 で新しいオペレーション `Upload Part (Copy)` が加えられて、マルチパートアップロード API が拡張されました。このマルチパートアップロード操作を使用して、最大 5 TB のサイズのオブジェクトをコピーできます。詳細については、「オブジェクトのコピー、移動、名前の変更」を参照してください。 マルチパートアップロード API の概念については、マルチパートアップロードを使用したオブジェクトのアップロードとコピー を参照してください。	2011 年 21 月 6 日

変更	説明	日付
HTTP を介した SOAP API 呼び出しの無効化	セキュリティの向上のため、HTTP を介した SOAP API 呼び出しが無効になりました。認証済みおよび匿名 SOAP リクエストは SSL を使用して、Amazon S3 に送信される必要があります。	2011 年 6 月 6 日
IAM によるクロスアカウント委任の有効化	これまでは、Amazon S3 リソースにアクセスするには、IAM ユーザーは親 AWS アカウントと Amazon S3 リソース所有者の両方からの許可が必要でした。アカウント間のアクセスによって、IAM ユーザーは所有者アカウントからのアクセス許可のみが必要になりました。つまり、リソース所有者が AWS アカウントへのアクセス許可を付与した場合、その AWS アカウントは IAM ユーザーにこれらのリソースへのアクセス許可を付与できるようになりました。 詳細については、IAM ユーザーガイドの「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。 バケットポリシーでのプリンシパルの指定については、バケットポリシーのプリンシパル を参照してください。	2011 年 6 月 6 日
新しいリンク	このサービスのエンドポイント情報は、現在、AWS 全般のリファレンスで参照できます。詳細については、AWS の全般的なリファレンスの「リージョンとエンドポイント」を参照してください。	2011 年 3 月 1 日

変更	説明	日付
Amazon S3 での静的ウェブサイトのホストのサポート	Amazon S3 に、静的ウェブサイトをホストするための拡張サポートが導入されました。これには、インデックスドキュメントとカスタムエラードキュメントのサポートが含まれます。これらの機能を使用すると、バケットのルートまたはサブフォルダへのリクエスト (例えば `http://mywebsite.com/subfolder`) は、バケット内のオブジェクトのリストの代わりに、インデックスドキュメントを返します。エラーが検出されると、Amazon S3 は Amazon S3 エラーメッセージの代わりにカスタムエラーメッセージを返します。詳細については、「Amazon S3 を使用して静的ウェブサイトをホスティングする」を参照してください。	2011 年 6 月 6 日
このサービスのエンドポイント情報は、現在、AWS 全般のリファレンスで参照できます。詳細については、AWS の全般的なリファレンスの「リージョンとエンドポイント」を参照してください。	2011 年 3 月 1 日	

変更	説明	日付
Amazon S3 での静的ウェブサイトのホストのサポート	Amazon S3 に、静的ウェブサイトをホストするための拡張サポートが導入されました。これには、インデックスドキュメントとカスタムエラードキュメントのサポートが含まれます。これらの機能を使用すると、バケットのルートまたはサブフォルダへのリクエスト (例えば `http://mywebsite.com/subfolder`) は、バケット内のオブジェクトのリストの代わりに、インデックスドキュメントを返します。エラーが検出されると、Amazon S3 は Amazon S3 エラーメッセージの代わりにカスタムエラーメッセージを返します。詳細については、「Amazon S3 を使用して静的ウェブサイトをホスティングする」を参照してください。	2011 年 2 月 17 日
レスポンスヘッダー API のサポート	GET Object REST API により、各リクエストの REST GET Object リクエストのレスポンスヘッダーを変更できるようになりました。つまり、オブジェクト自体を変更せずに、レスポンスのオブジェクトメタデータを変更できます。詳細については、「オブジェクトのダウンロード」を参照してください。	2011 年 1 月 14 日
大容量オブジェクトのサポート	Amazon S3 で S3 バケットに保存できるオブジェクトの最大サイズが5 GB から5 TB に増加しました。REST API を使用する場合、単一の PUT 操作で、最大 5 GB のオブジェクトをアップロードできます。これよりも大きなオブジェクトは、マルチパートアップロード REST API を使用して、オブジェクトを分割してアップロードする必要があります。詳細については、「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。	2010 年 12 月 9 日
マルチパートアップロード	マルチパートアップロードにより、Amazon S3 に高速で柔軟にアップロードできます。一連のパーツとして、単一のオブジェクトをアップロードできます。詳細については、「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。	2010 年 11 月 10 日

変更	説明	日付
バケットポリシーでの正規化 ID のサポート	バケットポリシーで正規化 ID を指定できるようになりました。詳細については、「バケットポリシーのプリンシパル」を参照してください。	2010 年 9 月 17 日
Amazon S3 の IAM との連携	このサービスが AWS Identity and Access Management (IAM) と連動するようになりました。詳細については、IAM ユーザーガイドの「IAM と連携する AWS のサービス」を参照してください。	2010 年 9 月 2 日
通知	Amazon S3 通知機能により、Amazon S3 がバケットで主要イベントを検出した場合に、Amazon Simple Notification Service (Amazon SNS) トピックにメッセージを公開するように、バケットを設定できます。詳細については、「Amazon S3 イベント通知の設定」を参照してください。	2010 年 7 月 14 日
バケットポリシー	バケットポリシーは、バケット、オブジェクト、一連のオブジェクト間でアクセス許可を設定するために使用するアクセス管理システムです。この機能は、アクセスコントロールリストを補完し、多くの場合に置き換わるものです。詳細については、「Amazon S3 のバケットポリシー」を参照してください。	2010 年 7 月 6 日
すべてのリージョンで使用可能なパススタイル構文	Amazon S3 では、米国クラシックリージョンの、またはバケットがリクエストのエンドポイントと同じリージョンにある場合、すべてのバケットでパススタイル構文をサポートするようになりました。詳細については、「バケットの仮想ホスティング」を参照してください。	2010 年 6 月 9 日
欧州 (アイルランド) の新しいエンドポイント	Amazon S3 は、欧州 (アイルランド) のエンドポイントを提供するようになりました: `http://s3-eu-west-1.amazonaws.com` 。	2010 年 6 月 9 日

変更	説明	日付
コンソール	AWS Management Console 経由で、Amazon S3 を使用できるようになりました。Amazon Simple Storage Service ユーザーガイドで、コンソールのすべての Amazon S3 機能について読むことができます。	2010 年 6 月 9 日
低冗長化	低冗長化ストレージに Amazon S3 のオブジェクトを格納することにより、Amazon S3 でストレージコストの削減を実現できます。詳細については、「低冗長化ストレージ」を参照してください。	2010 年 5 月 12 日
新しくサポートされるリージョン	Amazon S3 でアジアパシフィック (シンガポール) リージョンがサポートされるようになりました。詳細については、「バケットとリージョン」を参照してください。	2010 年 4 月 28 日
オブジェクトのバージョニング	本リリースでオブジェクトバージョニングが導入されました。すべてのオブジェクトはキーとバージョンを持つことができます。バケットのバージョニングを有効にすると、Amazon S3 はバケットに追加されたすべてのオブジェクトに一意のバージョン ID を与えます。この機能により、意図しない上書きと削除からの復元が可能です。詳細については、「バージョニング」と「バージョニングの使用」を参照してください。	2010 年 2 月 8 日
新しくサポートされるリージョン	Amazon S3 で米国西部 (北カリフォルニア) リージョンがサポートされるようになりました。このリージョンへのリクエストのための新しいエンドポイントは、`s3-us-west-1.amazonaws.com` です。詳細については、「バケットとリージョン」を参照してください。	2009 年 12 月 2 日

変更	説明	日付
AWS SDK for .NET	AWS では、REST や SOAP の代わりに、.NET 言語固有の API オペレーションを使用してアプリケーションを構築することを好むソフトウェアデベロッパー向けに、ライブラリ、サンプルコード、チュートリアル、その他のリソースを提供するようになりました。これらのライブラリはリクエスト認証、リクエストの再試行、エラー処理などの基本機能 (REST または SOAP API に含まれない) を提供しているため、簡単に開始できます。言語固有のライブラリおよびリソースについては、AWS SDK を使用した Amazon S3 での開発 を参照してください。	2009 年 11 月 11 日

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。