aws

開発者ガイド

Amazon CloudFront

Amazon CloudFront: 開発者ガイド

Table of Contents

Amazon CloudFront とは何ですか?

Amazon CloudFront は、ユーザーへの静的および動的なウェブコンテンツ (.html、.css、.js、イメージファイルなど) の配信を高速化するウェブサービスです。CloudFront では、エッジロケーションというデータセンターの世界的ネットワークを経由してコンテンツを配信します。CloudFront でサービスを提供しているコンテンツをユーザーがリクエストすると、リクエストはエッジロケーションにルーティングされ、レイテンシー (遅延時間) が最小になります。これにより、コンテンツは可能な限り最高のパフォーマンスで配信されます。

- コンテンツがエッジロケーション内に最も低いレイテンシーですでに存在している場合、CloudFront はそのコンテンツを即時に配信します。
- コンテンツがこのエッジロケーションに存在しない場合、CloudFront は、コンテンツの最終バージョンのソースとして識別されている Amazon S3 バケット、MediaPackage チャネル、または HTTP サーバー (たとえば、ウェブサーバー) などの定義されたオリジンからコンテンツを取り込みます。

たとえば、イメージが CloudFront からではなく従来のウェブサーバーから供給されているとします。たとえば、sunsetphoto.png というイメージを、URL `https://example.com/sunsetphoto.png` を使用して提供するとします。

ユーザーは簡単にこの URL にアクセスしてそのイメージを表示できます。ただし、そのイメージが見つかるまで、リクエストがネットワークから別のネットワークに (インターネットを構成する相互接続ネットワークの複雑な集合経由で) ルーティングされたということを、おそらくユーザーは認識しません。

CloudFront は、コンテンツを最良の方法で供給できるエッジロケーションに各ユーザーリクエストを AWS バックボーンネットワーク経由でルーティングすることで、コンテンツの配信を高速化します。通常、これはビューワーに最も高速に配信できる CloudFront エッジサーバーです。AWS ネットワークを使用することでユーザーのリクエストが通過しなければならないネットワークの数が大幅に減少するので、パフォーマンスが向上します。ユーザーが経験するレイテンシー (ファイルの最初のバイトがロードされるまでの時間) が低くなり、データ転送速度が高くなります。

お客様のファイル (オブジェクトとしても知られる) のコピーが世界中の複数のエッジロケーションに保持される (つまりキャッシュされる) ので、信頼性と可用性の向上も得られます。

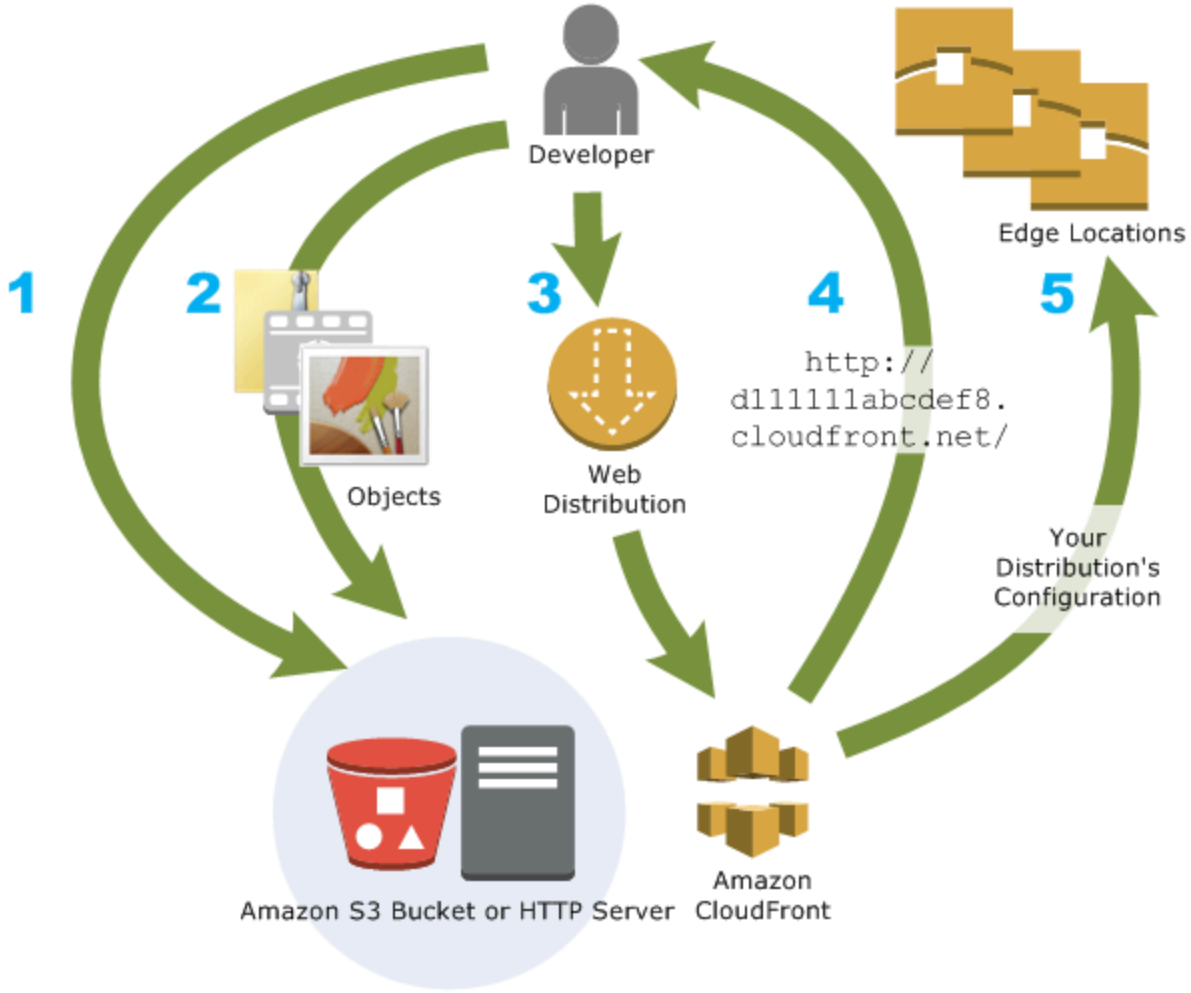
トピック

- コンテンツを配信するように CloudFront を設定する方法

- 料金
- CloudFront の使用方法
- CloudFront がコンテンツを配信する方法
- CloudFront エッジサーバーの場所と IP アドレス範囲
- AWS SDK での CloudFront の使用
- CloudFront テクニカルリソース

コンテンツを配信するように CloudFront を設定する方法

CloudFront ディストリビューションを作成して、コンテンツを配信する場所と、コンテンツ配信の追跡と管理の方法の詳細を CloudFront に指示します。こうすることで、誰かがそのコンテンツを表示または使用する際に、CloudFront はビューワーに近いコンピュータ (エッジサーバー) を使用してそのコンテンツをすばやく配信します。



コンテンツを配信するように CloudFront を構成する方法

1. Amazon S3 バケットや独自の HTTP サーバーなどのオリジンサーバーを指定します。ここから CloudFront がファイルを取得し、CloudFront エッジロケーションから全世界に配信します。

   オリジンサーバーには、お客様のオブジェクトのオリジナルの最終バージョンが保存されます。コンテンツを HTTP 経由で提供する場合、オリジンサーバーは Amazon S3 バケットまたは HTTP サーバー (ウェブサーバーなど) になります。HTTP サーバーは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、またはお客様が管理するサーバー (カスタムオリジンとも呼ばれる) で実行できます。

2. ファイルをオリジンサーバーにアップロードします。ファイル (オブジェクトとも呼ばれます) には、通常、ウェブページ、イメージ、メディアファイルが含まれますが、HTTP 経由で提供できるものであれば何でもかまいません。

   Amazon S3 バケットをオリジンサーバーとして使用している場合は、バケット内のオブジェクトを読み取り可能にして公開することで、オブジェクトの CloudFront URL を知っているユーザーなら誰でもそのオブジェクトにアクセスできます。オブジェクトを非公開にして、オブジェクトにアクセスするユーザーを制限することもできます。「署名付き URL と署名付き Cookie を使用したプライベートコンテンツを提供する」を参照してください。

3. お客様は CloudFront ディストリビューションを作成します。このディストリビューションにより、ユーザーがウェブサイトまたはアプリケーションを通じてファイルをリクエストしたときに、どのオリジンサーバーからファイルを取得するかが CloudFront に指示されます。同時に、詳細も指定します。たとえば、CloudFront ですべてのリクエストをログに記録するかどうかや、ディストリビューションを作成した直後にディストリビューションを有効にするかどうかなどです。

4. CloudFront は新しいディストリビューションにドメイン名を割り当てます。このドメイン名は、CloudFront コンソールで確認でき、プログラムによるリクエスト (API リクエストなど) ではレスポンスとして返されます。必要に応じて、代わりに使用する代替ドメイン名を追加できます。

5. CloudFront はディストリビューションの構成を関係するすべてのエッジロケーションまたはポイントオブプレゼンス (POP) (CloudFront がファイルのコピーをキャッシュするために使用する、地理的に分散して配置されたデータセンター内のサーバー群) に送信します。ただし、ディストリビューションのコンテンツは送信されません。

ウェブサイトやアプリケーションを開発するときには、CloudFront がお客様の URL に提供するドメイン名を使用します。たとえば、CloudFront がディストリビューションのドメ

イン名として d111111abcdef8.cloudfront.net を返した場合、Amazon S3 バケット内 (または HTTP サーバーのルートディレクトリ内) の logo.jpg の URL は https://d111111abcdef8.cloudfront.net/logo.jpg になります。

または、独自のドメイン名をディストリビューションに使用するよう CloudFront をセットアップすることもできます。この場合、URL は https://www.example.com/logo.jpg のようになります。

オプションで、ファイルにヘッダーを追加するようにオリジンサーバーを構成し、ファイルが CloudFront エッジロケーションのキャッシュに保持される期間を示すこともできます。デフォルトでは、各ファイルはエッジロケーションに 24 時間保持された後に有効期限切れになります。最小の有効期限切れ時間は 0 秒です。有効期限切れ時間の上限はありません。詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

料金

CloudFront では、エッジロケーションからのデータ転送と、HTTP または HTTPS リクエストに対して課金されます。料金は、使用タイプ、地域、選択した機能によって異なります。

Amazon Simple Storage Service (Amazon S3)、Elastic Load Balancing、Amazon API Gateway などの AWS オリジンを使用すると、オリジンから CloudFront へのデータ転送は常に無料です。AWS オリジンを使用した場合、CloudFront からビューワーへのアウトバウンドデータ転送にのみ課金されます。

詳細については、「CloudFront の料金」と、請求および Savings Bundle の「よくある質問」を参照してください。

CloudFront の使用方法

CloudFront を使用すると、さまざまな目標を達成するのに役立ちます。このセクションでは、詳細情報へのリンクとともに、可能性についてのアイデアをいくつか示します。

トピック

- 静的ウェブサイトのコンテンツ配信の加速
- オンデマンドビデオおよびライブストリーミングビデオの配信
- システム処理全体で特定のフィールドを暗号化する
- エッジのカスタマイズ

- Lambda@Edge カスタマイズを使用したプライベートコンテンツの供給

静的ウェブサイトのコンテンツ配信の加速

CloudFront は、世界中のビューワーへの静的コンテンツ (たとえば、イメージ、スタイルシート、JavaScript など) の配信を高速化します。CloudFront を使用することで、AWS バックボーンネットワークと CloudFront エッジサーバーの利点を活用して、ウェブサイトを閲覧する訪問者に高速かつ安全で、信頼性に優れたエクスペリエンスを提供できます。

Amazon S3 バケットを使用すると、静的コンテンツを簡単に格納し、配信することができます。CloudFront と S3 を組み合わせて使用することには、簡単に S3 コンテンツへのアクセスを制限できるオリジンアクセスコントロールを使用するオプションを含め、多くの利点があります。

使用をすばやく開始するために役立つ AWS CloudFormation テンプレートを含めた、CloudFront と S3 を併用する方法の詳細については、「Amazon S3 + Amazon CloudFront: A Match Made in the Cloud」を参照してください。

オンデマンドビデオおよびライブストリーミングビデオの配信

CloudFront では、録画済みのファイルとライブイベントの両方のメディアを世界中のビューワーにストリーム配信するためのいくつかのオプションが提供されています。

- オンデマンドビデオ (VOD) ストリーミングで CloudFront を使用して、MPEG DASH、Apple HLS、Microsoft Smooth Streaming、CMAF などの一般的な形式であらゆるデバイスにストリーミングできます。
- ライブストリームをブロードキャストする場合は、フラグメントを正しい順序で配信するマニフェストファイルに対する複数のリクエストを結合して、オリジンサーバーの負荷を軽減できるように、メディアフラグメントをエッジにキャッシュすることができます。

CloudFront でストリーミングコンテンツを配信する方法の詳細については、「CloudFront を使用したビデオオンデマンドおよびライブストリーミングビデオ」を参照してください。

システム処理全体で特定のフィールドを暗号化する

CloudFront を使用して HTTPS を設定する場合、すでにオリジンサーバーへのセキュアなエンドツーエンド接続が存在します。フィールドレベルの暗号化を追加する場合、HTTPS セキュリティに加えて、システムの処理中に特定のデータにオリジンの特定のアプリケーションのみがアクセスできるように、そのデータを保護できます。

フィールドレベルの暗号化を設定するには、CloudFront にパブリックキーを追加し、そのキーを使用して暗号化する一連のフィールドを指定します。詳細については、「フィールドレベル暗号化を使用した機密データの保護」を参照してください。

エッジのカスタマイズ

エッジでサーバーレスコードを実行することには、ビューワーのコンテンツとエクスペリエンスをカスタマイズして、待ち時間を短縮することができる様々な可能性があります。たとえば、メンテナンスのためにオリジンサーバーがダウンしているときにカスタムエラーメッセージを表示することができるため、ビューワーには一般的な HTTP エラーメッセージは表示されません。または、CloudFront がリクエストをオリジンに転送する前に、コンテンツにアクセスするユーザーを認証および管理する関数を使用できます。

Lambda@Edge と CloudFront を併用すると、CloudFront が配信するコンテンツをさまざまな方法でカスタマイズできます。Lambda@Edge の詳細、および CloudFront で関数を作成してデプロイする方法の詳細については、「Lambda@Edge を使用してエッジでカスタマイズする」を参照してください。独自のソリューションのためにカスタマイズできる多くのコードサンプルについては、Lambda@Edge 関数の例 を参照してください。

Lambda@Edge カスタマイズを使用したプライベートコンテンツの供給

Lambda@Edge を使用すると、署名付き URL や署名付き Cookie の使用に加えて、独自のカスタムオリジンからプライベートコンテンツを供給する CloudFront ディストリビューションを設定するのに役立ちます。

CloudFront を使用してこのプライベートコンテンツを供給するには、次のことを実行します。

- 署名付き URL または署名付き Cookie を使用してコンテンツにアクセスするようにユーザー (ビューワー) にリクエストします。
- オリジンへのアクセスを制限して、CloudFront のオリジン向けサーバーからのみ利用できるようにします。これは、次のいずれかで実行できます。
  - Amazon S3 オリジンでは、オリジンアクセスコントロール (OAC) を使用できます。
  - カスタムオリジンでは、次のように実行できます。
    - カスタムオリジンが Amazon VPC セキュリティグループまたは AWS Firewall Manager によって保護されている場合、CloudFront マネージドプレフィックスリストを使用して、CloudFront のオリジン向け IP アドレスからのみ、オリジンへのインバウンドトラフィックを許可できます。

- カスタム HTTP ヘッダーを使用して、CloudFront からのリクエストのみにアクセスを制限します。詳細については、the section called “カスタムオリジンのファイルへのアクセスを制限する”およびthe section called “オリジンリクエストにカスタムヘッダーを追加する”を参照してください。カスタムヘッダーを使用して Application Load Balancer オリジンへのアクセスを制限する例については、the section called “Application Load Balancer へのアクセスを制限する” を参照してください。
- カスタムオリジンにカスタムアクセス制御ロジックが必要な場合は、このブログの記事、Amazon CloudFront と Lambda@Edge を使用してプライベートコンテンツを供給するで説明されているように、Lambda@Edge を使用してそのロジックを実装できます。

CloudFront がコンテンツを配信する方法

初期セットアップを行った後、CloudFront は、ウェブサイトまたはアプリケーションと連携して動作し、コンテンツの配信を高速化します。このセクションでは、ビューワーがコンテンツをリクエストしたときに CloudFront がコンテンツを配信する方法について説明します。

トピック

CloudFront がユーザーにコンテンツを配信する方法

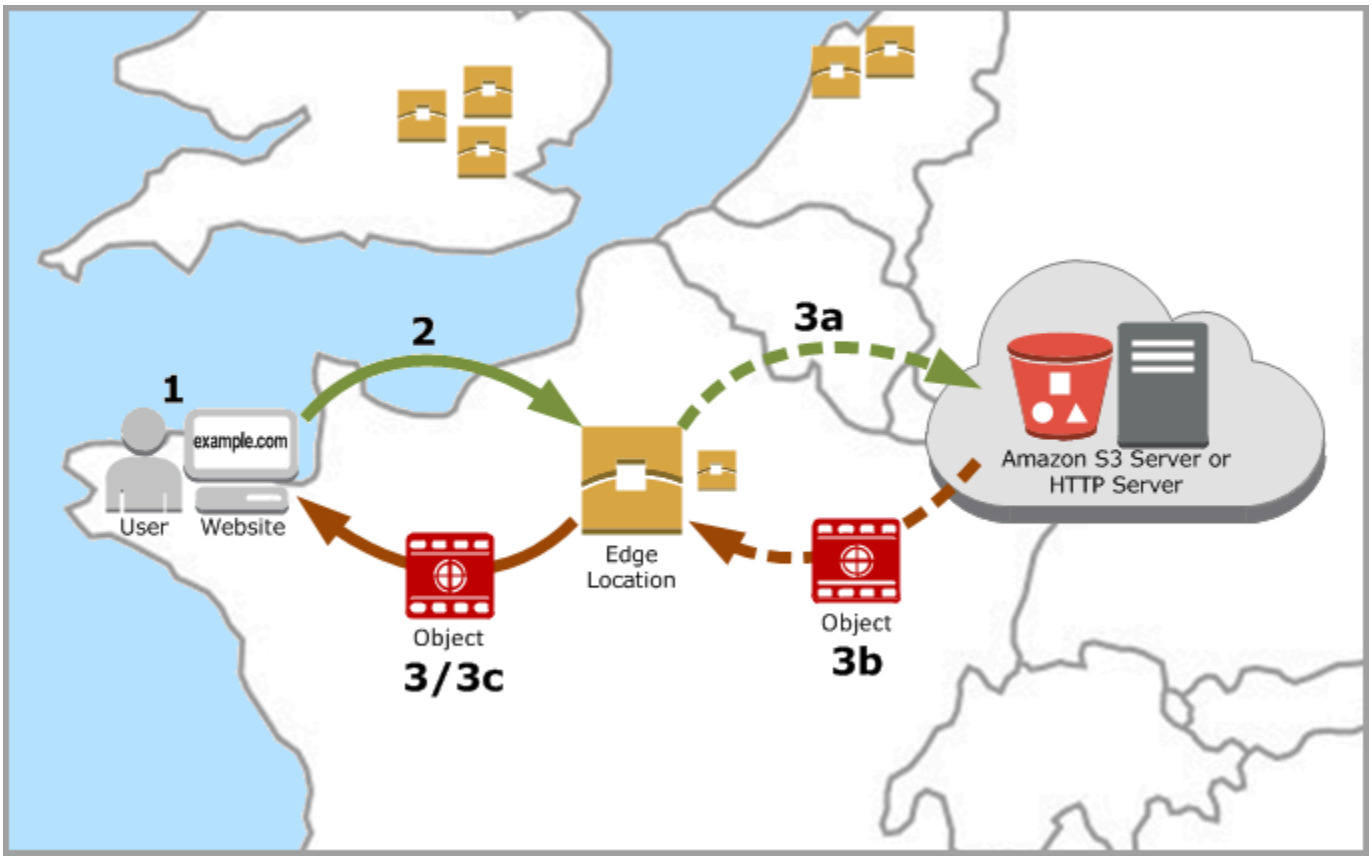
CloudFront でコンテンツ配信を設定すると、ユーザーがオブジェクトをリクエストしたときに次の処理が行われます。

1. ユーザーがウェブサイトまたはアプリケーションにアクセスして、イメージファイルや HTML ファイルなどのオブジェクトに対するリクエストを送信します。
2. DNS は、リクエストを、最も適切に処理できる CloudFront POP (エッジロケーション) にルーティングします。通常は、レイテンシーを考慮して最寄りの CloudFront POP が対象となります。
3. CloudFront では、リクエストされたオブジェクトのキャッシュをチェックします。オブジェクトがキャッシュにある場合、CloudFront はオブジェクトをユーザーに返します。オブジェクトがキャッシュにない場合、CloudFront は次の処理を実行します。

a. CloudFront は、リクエストを、ディストリビューションで指定されている内容と照合し、ファイルのリクエストを、対応するオブジェクトに応じてオリジンサーバーに転送します (例えば、Amazon S3 バケットや HTTP サーバー)。

b. そのオリジンサーバーが、エッジロケーションにオブジェクトを返します。

c. オリジンから最初のバイトが到着した直後に、CloudFront はそのオブジェクトをユーザーに転送し始めます。また、CloudFront は、そのオブジェクトに対する次回のリクエストに備えて、キャッシュにそのオブジェクトを追加します。



CloudFront とリージョン別エッジキャッシュとの連携

CloudFront ポイントオブプレゼンス (POP またはエッジロケーションとも呼ばれます) を使用することで、人気のあるコンテンツをすばやくビューワーに提供できます。CloudFront にはリージョン別エッジキャッシュもあり、コンテンツが POP に残るだけの人気が十分にない場合でも、より多くのコンテンツをビューワーの近くに配置して、そのコンテンツのパフォーマンスを向上させます。

リージョン別エッジキャッシュは、すべてのタイプのコンテンツ (特に、時間の経過とともに人気が落ちる傾向にあるコンテンツ) に役立ちます。この例には、ビデオ、写真、アートワークのような

ユーザーが生成したコンテンツ、製品の写真やビデオのような e コマースアセット、突然新たに人気が出る可能性がある二ュースやイベント関連のコンテンツがあります。

リージョン別キャッシュの動作

リージョン別エッジキャッシュは、ビューワーに近接して世界各地にデプロイされる CloudFront ロケーションです。オリジンサーバーと、ビューワーに直接コンテンツを提供する POP (世界各地のエッジロケーション) の間にあります。オブジェクトの人気が下がると、個別の POP では、これらのオブジェクトを削除し、より人気の高いコンテンツ用に容量を確保する場合があります。リージョン別エッジキャッシュのキャッシュは個別の POP よりも大きいため、オブジェクトは最も近いリージョン別エッジキャッシュロケーションでより長くキャッシュに残ります。これにより、より多くのコンテンツがビューワーの近くに保持されるため、CloudFront がオリジンサーバーに戻る必要がなくなり、ビューワーに対する全般的なパフォーマンスが向上します。

ビューワーがウェブサイトで、またはアプリケーション経由でリクエストを実行すると、DNS はユーザーのリクエストに対応できる最適な POP にリクエストをルーティングします。通常、この場所は、レイテンシーに関して最も近い CloudFront エッジロケーションです。その POP では、CloudFront によってリクエストされたオブジェクトがキャッシュにあるかどうかをチェックします。オブジェクトがキャッシュにある場合、CloudFront はオブジェクトをユーザーに返します。オブジェクトがキャッシュにない場合、POP は最も近いリージョン別エッジキャッシュをフェッチします。POP がリージョン別エッジキャッシュをスキップして直接オリジンに移動する場合の詳細については、次の注記を参照してください。

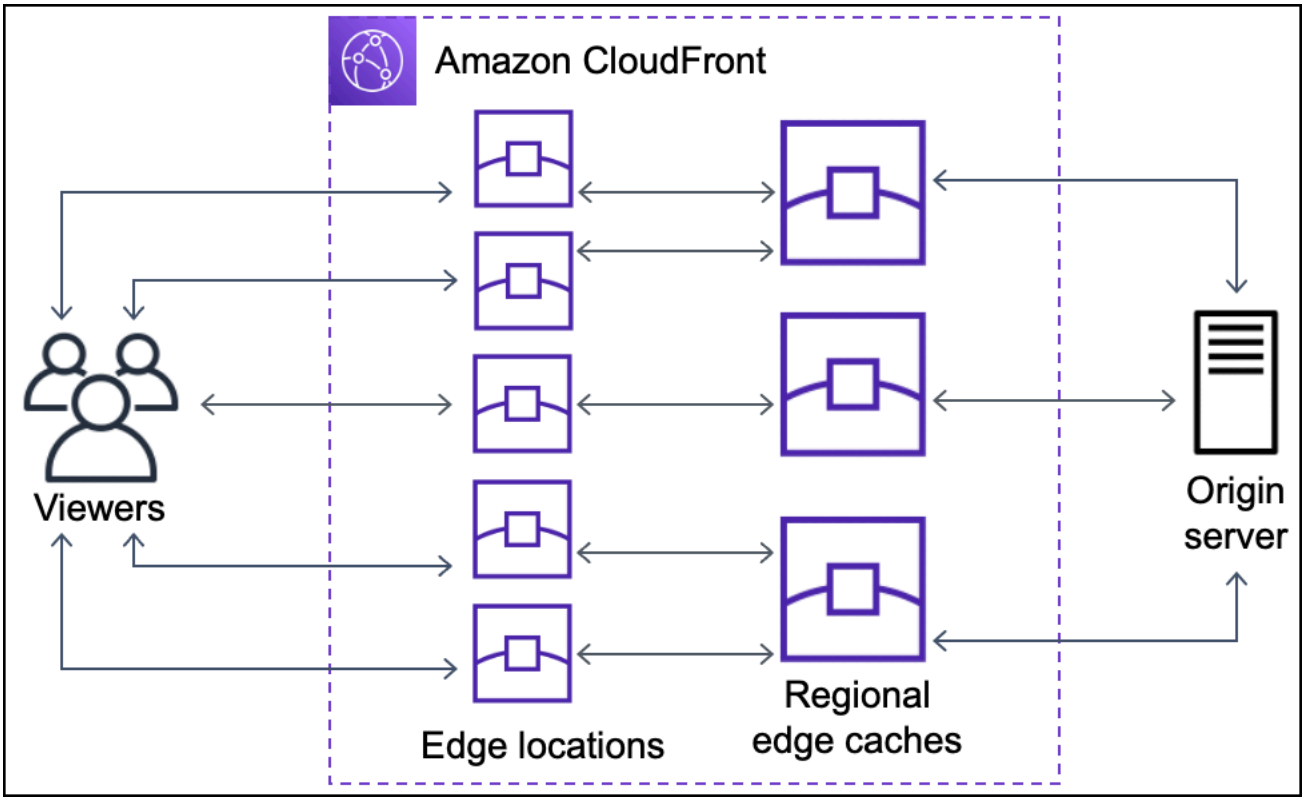
そのリージョン別エッジロケーションで、CloudFront は要求されたオブジェクトがキャッシュにあるかどうかをもう一度チェックします。オブジェクトがキャッシュにある場合、CloudFront はリクエスト元の POP にオブジェクトを転送します。リージョン別エッジキャッシュロケーションから最初のバイトが到着した直後に、CloudFront はそのオブジェクトをユーザーに転送し始めます。また、CloudFront は、そのオブジェクトに対する次回のリクエストに備えて、POP 内のキャッシュにそのオブジェクトを追加します。

POP とリージョン別エッジキャッシュロケーションのいずれかでキャッシュされていないオブジェクトについて、CloudFront はディストリビューションの仕様とリクエストを比較し、リクエストをオリジンサーバーに転送します。オリジンサーバーがオブジェクトをリージョン別エッジキャッシュロケーションに送り返すと、POP に転送され、CloudFront はオブジェクトをユーザーに転送します。この場合、CloudFront は次回にビューワーがオブジェクトをリクエストしたときに、POP に加えてリージョン別エッジキャッシュロケーションのキャッシュにオブジェクトを追加します。これにより、リージョン内のすべての POP がローカルキャッシュを共有されて、オリジンサーバーに対して複数のリクエストがおこなわれなくなります。また、CloudFront はオリジンサーバーと永続的接続を保持しているため、可能な限り迅速にオリジンからオブジェクトが取得されます。

Note

- リージョン別エッジキャッシュには、POP と同等の機能があります。たとえば、キャッシュ無効化リクエストでは、有効期限が切れる前に、POP とリージョン別エッジキャッシュの両方からオブジェクトが削除されます。エンドユーザーが次にオブジェクトを要求したときに、CloudFront はオリジンに戻ってオブジェクトの最新バージョンをフェッチします。
- プロキシ HTTP メソッド (PUT、POST、PATCH、OPTIONS、DELETE) は POP からオリジンに直接送信され、リージョン別エッジキャッシュを経由してプロキシを実行しません。
- リクエスト時に決定される動的リクエストは、リージョン別エッジキャッシュを通過せず、直接オリジンに送信されます。
- オリジンが Amazon S3 バケットで、リクエストの最適なリージョン別エッジキャッシュが S3 バケットと 同じ AWS リージョン である場合、POP はリージョン別エッジキャッシュをスキップして S3 バケットに直接移動します。

次の図は、CloudFront エッジロケーションとリージョン別エッジキャッシュを通るリクエストとレスポンスの流れを示しています。



CloudFront エッジサーバーの場所と IP アドレス範囲

CloudFront エッジサーバーの場所の一覧については、「Amazon CloudFront グローバルエッジネットワーク」をご覧ください。

Amazon Web Services (AWS) は、その現在の IP アドレス範囲を JSON 形式で公開します。現在の範囲を参照するには、ip-ranges.json ファイルをダウンロードします。詳細については、Amazon Web Services 全般のリファレンス の AWS IP アドレスの範囲をご参照ください。

CloudFront エッジサーバーに関連付けられた IP アドレス範囲を見つけるには、ip-ranges.json ファイルで次の文字列を検索します。

```
"region": "GLOBAL",
"service": "CLOUDFRONT"
```

また、https://d7uri8nf7uskq.cloudfront.net/tools/list-cloudfront-ips での CloudFront の IP 範囲についてのみ表示できます。

CloudFront マネージドプレフィックスリストを使用

CloudFront マネージドプレフィクスリストには、CloudFront のグローバルに分散されたオリジン向けサーバーの IP アドレス範囲が含まれています。オリジンが AWS でホストされ、AmazonVPC セキュリティグループで保護されている場合、CloudFront マネージドプレフィックスリストを使用して、CloudFront のオリジン側サーバーからのみオリジンへのインバウンドトラフィックを許可し、CloudFront 以外のトラフィックがオリジンに到達するのを防ぐことができます。CloudFront がマネージドプレフィクスリストを維持するため、CloudFront のグローバルオリジン向けサーバーの IP アドレスは常に最新になります。CloudFront マネージドプレフィクスリストを使用すると、IP アドレス範囲のリストを自分で読み取ったり、管理したりする必要はなくなります。

例えば、オリジンが欧州 (ロンドン) (eu-west-2) リージョンの Amazon EC2 インスタンスであると想定します。インスタンスが VPC 内にある場合は、CloudFront マネージドプレフィクスリストからのインバウンド HTTPS アクセスを許可するセキュリティグループルールを作成できます。これにより、CloudFront のグローバルオリジン向けのすべてのサーバーがインスタンスに到達できるようになります。セキュリティグループから他のすべてのインバウンドルールを削除すると、CloudFront 以外のトラフィックがインスタンスに到達するのを防ぐことができます。

CloudFront マネージドプレフィクスリストの名前は com.amazonaws.global.cloudfront.origin-facing です。詳細については、Amazon VPC ユーザーガイドの「AWS マネージドプレフィックスリストの使用」を参照してください。

> **Important**
>
> CloudFront マネージドプレフィックスリストは、Amazon VPC クォータに適用される点で独特です。詳細については、Amazon VPC ユーザーガイドの「AWS マネージドプレフィックスリストの重み」を参照してください。

AWS SDK での CloudFront の使用

AWS ソフトウェア開発キット (SDK) は、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コードの例
AWS SDK for C++	AWS SDK for C++ コードの例
AWS CLI	AWS CLI コードの例
AWS SDK for Go	AWS SDK for Go コードの例
AWS SDK for Java	AWS SDK for Java コードの例
AWS SDK for JavaScript	AWS SDK for JavaScript コードの例
AWS SDK for Kotlin	AWS SDK for Kotlin コードの例
AWS SDK for .NET	AWS SDK for .NET コードの例
AWS SDK for PHP	AWS SDK for PHP コードの例
AWS Tools for PowerShell	Tools for PowerShell のコード例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コードの例
AWS SDK for Ruby	AWS SDK for Ruby コードの例
AWS SDK for Rust	AWS SDK for Rust コードの例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コードの例

SDK ドキュメント	コードの例
AWS SDK for Swift	AWS SDK for Swift コードの例

> **可用性の例**
>
> 必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

CloudFront テクニカルリソース

以下のリソースを使用して、CloudFront に関する技術的な質問に対する回答を取得します。

- AWS re:Post – デベロッパーが CloudFront に関連する技術的な質疑応答を行える、デベロッパー向けのコミュニティサイト。
- AWS Support センター – このサイトには、最近のサポートケースと、AWS Trusted Advisor およびヘルスチェックの結果に関する情報が含まれています。また、ディスカッション フォーラム、技術的な FAQ、サービスヘルスダッシュボード、および AWS Support プランに関する情報へのリンクも提供します。
- AWS プレミアムサポート - 1 対 1 での迅速な対応を行うサポートチャネルである AWS プレミアムサポートに関して説明します。AWS でのアプリケーションの構築および実行を支援します。
- AWS IQ – AWS 認定プロフェッショナルとエキスパートからのヘルプを得ます。

CloudFront の開始方法

このセクションのトピックでは、Amazon CloudFront を使用してコンテンツの配信を開始する方法について説明します。

「セットアップする」トピックでは、AWS アカウント の作成や、管理アクセス権を持つユーザーの作成など、以下のチュートリアルの前提条件について説明します。

基本的なディストリビューションのチュートリアルでは、認証されたリクエストを Amazon S3 オリジンに送信するようにオリジンアクセスコントロール (OAC) を設定する方法を示します。

安全な静的ウェブサイトのチュートリアルでは、Amazon S3 オリジンで OAC を使用してドメイン名の安全な静的ウェブサイトを作成する方法を示します。チュートリアルでは、設定とデプロイに Amazon CloudFront (CloudFront) テンプレートを使用します。

トピック

- セットアップする
- 基本的な CloudFront ディストリビューションの開始方法
- 安全な静的ウェブサイトの開始方法

セットアップする

このトピックでは、Amazon CloudFront を使用する前の事前ステップ (AWS アカウントの作成など) について説明します。

トピック

- AWS アカウントへのサインアップ
- 管理アクセスを持つユーザーを作成する
- CloudFront へのアクセス方法を選択する

AWS アカウントへのサインアップ

AWS アカウントがない場合は、以下のステップを実行して作成します。

AWS アカウントにサインアップするには

1. https://portal.aws.amazon.com/billing/signup を開きます。

2. オンラインの手順に従います。

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

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

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

管理アクセスを持つユーザーを作成する

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

AWS アカウントのルートユーザーをセキュリティで保護する

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

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

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

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

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

2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

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

管理アクセス権を持つユーザーとしてサインインする

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

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

追加のユーザーにアクセス権を割り当てる

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

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。
2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの参加」を参照してください。

CloudFront へのアクセス方法を選択する

Amazon CloudFront には、以下の方法でアクセスできます。

- AWS Management Console – このガイドの手順は、AWS Management Console を使用してタスクを実行する方法について説明します。
- AWS SDK – AWS が SDK を提供しているプログラミング言語を使用している場合は、SDK を使用して CloudFront にアクセスできます。SDK によって認証が簡素化され、開発環境との統合が容易になり、CloudFront コマンドにアクセスすることができます。詳細については、「AWS SDK での CloudFront の使用」を参照してください。

- CloudFront API – SDK が提供されていないプログラミング言語を使用している場合、API アクションの情報と API リクエストの作成方法については、Amazon CloudFront API リファレンスを参照してください。
- AWS CLI – AWS Command Line Interface (AWS CLI) は AWS のサービスを管理するための統合ツールです。AWS CLI をインストールして設定する方法については、「AWS Command Line Interface ユーザーガイド」の「AWS CLI の最新バージョンをインストールまたは更新する」を参照してください。
- Tools for Windows PowerShell – Windows PowerShell の使用経験がある場合は、AWS Tools for Windows PowerShell を使用することもできます。詳細については、「AWS Tools for Windows PowerShell ユーザーガイド」の「AWS Tools for Windows PowerShell のインストール」を参照してください。

基本的な CloudFront ディストリビューションの開始方法

このセクションの手順では、CloudFront を使用して、次のことを行う基本的な設定をセットアップする方法について説明します。

- ディストリビューションオリジンとして使用するバケットを作成します。
- オブジェクトの元のバージョンを Amazon Simple Storage Service (Amazon S3) バケットに保存します。
- オリジンアクセスコントロール (OAC) を使用して、認証済みのリクエストを Amazon S3 オリジンに送信します。OAC は CloudFront を介してリクエストを送信することで、ビューワーが S3 バケットに直接アクセスできないようにします。OAS の詳細については、「Amazon Simple Storage Service オリジンへのアクセスを制限する」を参照してください。
- オブジェクトの URL で CloudFront ドメイン名を使用します (例: `https://d111111abcdef8.cloudfront.net/index.html`)。
- オブジェクトを、デフォルトの 24 時間 (最小時間は 0 秒) にわたって、CloudFront エッジロケーションに保存します。

大半のオプションはカスタマイズ可能です。CloudFront ディストリビューションのオプションをカスタマイズする方法の詳細については、「ディストリビューションを作成する」を参照してください。

トピック

- ステップ 1: Amazon S3 バケットを作成する
- ステップ 2: コンテンツをバケットにアップロードする
- ステップ 3: OAC で Amazon S3 オリジンを使用する CloudFront ディストリビューションを作成する
- ステップ 4: CloudFront からコンテンツにアクセスする
- ステップ 5 : クリーンアップ
- 基本的な CloudFront ディストリビューションを強化する

前提条件

この作業を開始する前に、必ず「セットアップする」のステップを完了してください。

ステップ 1: Amazon S3 バケットを作成する

Amazon S3 バケットは、ファイル (オブジェクト) やフォルダのコンテナです。CloudFront では、S3 バケットがソースである場合、ほとんどのタイプのファイルを配信できます。たとえば、CloudFront では、テキスト、画像、ビデオを配信できます。Amazon S3 に保存できるデータ量に上限はありません。

このチュートリアルでは、基本的なウェブページを作成するために提供されたサンプルの `hello world` ファイルを使用して S3 バケットを作成します。

バケットを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. この「開始方法」では、Hello World サンプルを使用することをお勧めします。hello world ウェブページ (hello-world-html.zip) をダウンロードします。解凍して、`css` フォルダと `index` ファイルを使いやすい場所 (ブラウザを実行しているデスクトップなど) に保存します。
3. [バケットを作成] を選択します。
4. 「Amazon Simple Storage Service ユーザーガイド」の「汎用バケットの命名規則」に準拠した一意のバケット名を入力します。
5. [リージョン] では、地理的に近い AWS リージョンを選択することをお勧めします (これにより、レイテンシーが短縮され、コストが削減されます)。

    - 別のリージョンを選択することもできます。規制要件に対応する場合などは、別のリージョンを選択することがあります。

6. 他のすべての設定はデフォルトのままにして、[バケットを作成] を選択します。

ステップ 2: コンテンツをバケットにアップロードする

Amazon S3 バケットを作成したら、解凍した `hello world` ファイルの内容をバケットにアップロードします (このファイルは ステップ 1: Amazon S3 バケットを作成する にダウンロードして解凍しました)。

コンテンツを Amazon S3 にアップロードするには

1. [汎用バケット] セクションで、新しいバケットの名前を選択します。
2. [アップロード] を選択します。
3. [アップロード] ページで、`css` フォルダと `index` ファイルをドロップ領域内にドラッグします。
4. 他のすべての設定はデフォルトのままにして、[アップロード] を選択します。

ステップ 3: OAC で Amazon S3 オリジンを使用する CloudFront ディストリビューションを作成する

このチュートリアルでは、オリジンアクセスコントロール (OAC) で Amazon S3 オリジンを使用する CloudFront ディストリビューションを作成します。OAC は、認証済みのリクエストを Amazon S3 オリジンに安全に送信するのに役立ちます。OAC の詳細については、「Amazon Simple Storage Service オリジンへのアクセスを制限する」を参照してください。

OAC を使用する Amazon S3 オリジンで CloudFront ディストリビューションを作成するには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。
2. [ディストリビューションの作成] を選択します。
3. [オリジン] の [オリジンドメイン] で、このチュートリアル用に作成した S3 バケットを選択します。
4. [オリジン] の [オリジンアクセス] で、[オリジンアクセスコントロール設定 (推奨)] を選択します。
5. [オリジンアクセスコントロール] で、[新しい OAC を作成] を選択します。
6. [新しい OAC を作成] ペインは、デフォルト設定のままにして、[作成] を選択します。
7. [ウェブアプリケーションファイアウォール (WAF)] で、いずれかのオプションを選択します。

8. 他のすべてのセクションや設定は、デフォルト値のままにします。これらのパラメータの詳細については、「ディストリビューションの設定」を参照してください。
9. [ディストリビューションの作成] を選択します。
10. [S3 バケットポリシーを更新する必要があります] バナーで、メッセージを読み、[ポリシーをコピー] を選択します。
11. 同じバナーで、[S3 バケットのアクセス許可に移動してポリシーを更新する] へのリンクを選択します (これにより、Amazon S3 コンソールのバケットの詳細ページに移動します)。
12. [バケットポリシー] で、[編集] を選択します。
13. [ステートメントを編集] フィールドに、ステップ 10 でコピーしたポリシーを貼り付けます。
14. [Save changes] (変更の保存) をクリックします。
15. CloudFront コンソールに戻り、新しいディストリビューションの [詳細] セクションを確認します。ディストリビューションのデプロイが完了すると、[最終変更日] フィールドが [デプロイ中] から 日付と時刻に変わります。
16. CloudFront がディストリビューションに割り当てるドメイン名を記録します。この表示は以下のようになります: `d111111abcdef8.cloudfront.net`

このチュートリアルのディストリビューションと S3 バケットを本番環境で使用する前に、特定のニーズに合うように設定してください。本番環境でのアクセス権の設定方法については、「コンテンツへのセキュアなアクセスの設定とアクセスの制限」を参照してください。

ステップ 4: CloudFront からコンテンツにアクセスする

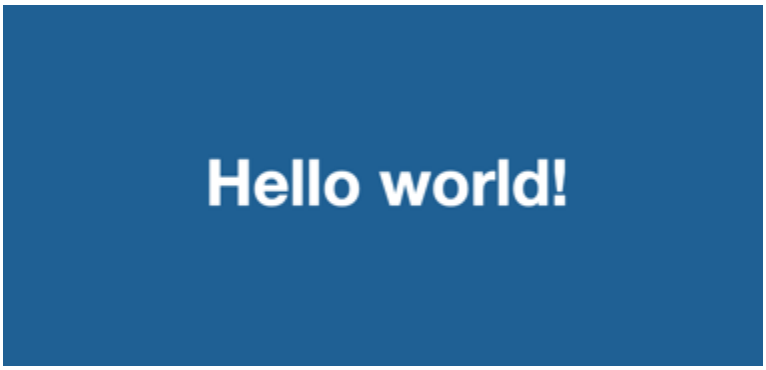
CloudFront からコンテンツにアクセスするには、CloudFront ディストリビューションのドメイン名と、コンテンツのメインページを組み合わせます (ディストリビューションのドメイン名は ステップ 3: OAC で Amazon S3 オリジンを使用する CloudFront ディストリビューションを作成する に記録しました)。

- ディストリビューションのドメイン名は、`d111111abcdef8.cloudfront.net` のようになります。
- 通常、ウェブサイトのメインページへのパスは `/index.html` です。

したがって、CloudFront からコンテンツにアクセスするための URL は次のようになります。

`https://d111111abcdef8.cloudfront.net/index.html`.

前の手順に従い、hello world ウェブページを使用している場合は、次のようにコンテンツが表示されます。



この S3 バケットに追加のコンテンツをアップロードした場合、CloudFront ディストリビューションのドメイン名と S3 バケット内のオブジェクトへのパスを組み合わせることで、CloudFront からコンテンツにアクセスできます。例えば、`new-page.html` という名前の新しいファイルを S3 バケットのルートにアップロードした場合、URL は次のようになります。

`https://d111111abcdef8.cloudfront.net/new-page.html`.

ステップ 5：クリーンアップ

学習目的のためだけにディストリビューションと S3 バケットを作成した場合は、削除して料金が発生しないようにします。最初にディストリビューションを削除します。詳細については、以下のリンクを参照してください。

- ディストリビューションを削除する
- バケットの削除

基本的な CloudFront ディストリビューションを強化する

この開始方法チュートリアルでは、ディストリビューションを作成するための最小限のフレームワークを提供します。以下の拡張機能を試してみることをお勧めします。

- デフォルトでは、Amazon S3 バケット内のファイル (オブジェクト) はプライベートとして設定されます。バケットを作成した AWS アカウントのみが、ファイルの読み取りまたは書き込みのアクセス許可を持ちます。自分の Amazon S3 バケット内のファイルに他のユーザーが CloudFront の URL を使用してアクセスできるようにするには、そのオブジェクトにパブリック読み取りアクセス許可を付与する必要があります。

- CloudFront のプライベートコンテンツ機能を使用して、Amazon S3 バケット内のコンテンツへのアクセスを制限できます。プライベートコンテンツの配信の詳細については、「署名付き URL と署名付き Cookie を使用したプライベートコンテンツを提供する」を参照してください。
- CloudFront ディストリビューションは、カスタムドメイン名 (d111111abcdef8.cloudfront.net の代わりに www.example.com など) を使用するように設定できます。詳細については、「カスタム URL を使用する」を参照してください。
- このチュートリアルでは、オリジンアクセスコントロール (OAC) で Amazon S3 オリジンを使用しています。ただし、オリジンがウェブサイトエンドポイントとして設定した S3 バケットである場合は、OAC を使用できません。この場合は、CloudFront でバケットをカスタムオリジンとして設定する必要があります。詳細については、「ウェブサイトのエンドポイントとして設定された Amazon S3 バケットを使用する」を参照してください。OAC の詳細については、「Amazon Simple Storage Service オリジンへのアクセスを制限する」を参照してください。

安全な静的ウェブサイトの開始方法

Amazon CloudFront の使用を開始するには、このトピックで説明するソリューションを使用して、ドメイン名用の安全な静的ウェブサイトを作成します。静的ウェブサイトは、HTML、CSS、JavaScript、画像、動画などの静的ファイルのみを使用し、サーバーやサーバー側の処理は必要ありません。このソリューションを使用すると、ウェブサイトには次の利点があります。

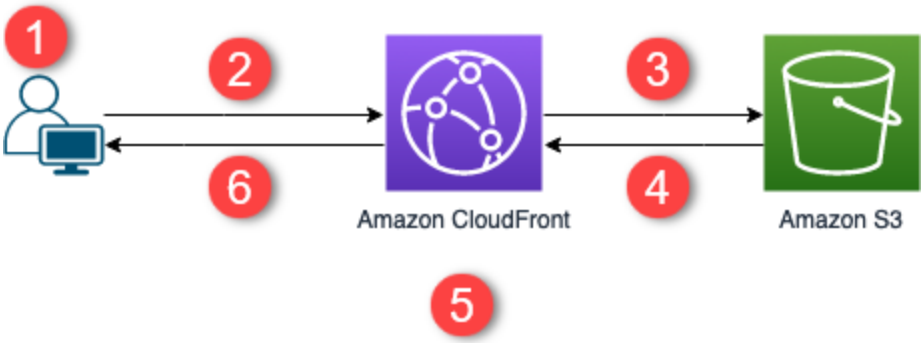
- Amazon Simple Storage Service (Amazon S3) の耐久性のあるストレージを使用 - このソリューションでは、静的ウェブサイトのコンテンツをホストする Amazon S3 バケットを作成します。ウェブサイトを更新するには、新しいファイルを S3 バケットにアップロードするだけです。
- Amazon CloudFront コンテンツ配信ネットワークによる高速化 - このソリューションは、低レイテンシーでビューワーにウェブサイトを提供する CloudFront ディストリビューションを作成します。このディストリビューションは、オリジンアクセスコントロール (OAC) を使用して設定され、S3 から直接ではなく CloudFront を通じてのみウェブサイトにアクセスできるようにします。
- HTTPS とセキュリティヘッダーによってセキュリティ保護される - このソリューションは、AWS Certificate Manager (ACM) で SSL/TLS 証明書を作成し、それを CloudFront ディストリビューションにアタッチします。この証明書により、ディストリビューションが HTTPS を使用してドメインの Web サイトに安全にサービスを提供できるようになります。
- AWS CloudFormation を使用した設定とデプロイ - このソリューションは AWS CloudFormation テンプレートを使用してすべてのコンポーネントをセットアップするため、コンポーネントの設定よりもウェブサイトのコンテンツに集中できます。

このソリューションは、GitHub 上のオープンソースです。コードを表示したり、プルリクエストを送信したり、問題を提起したりするには、「」に進みますhttps://github.com/aws-samples/amazon-cloudfront-secure-static-site

トピック

ソリューションの概要

次の図は、この静的ウェブサイトソリューションの動作の概要を示しています。



1. ビューワーが www.example.com のウェブサイトをリクエストします。
2. リクエストされたオブジェクトがキャッシュされている場合、CloudFront はオブジェクトをキャッシュからビューワーに返します。
3. オブジェクトが CloudFront のキャッシュにない場合、CloudFront はオリジン (S3 バケット) にオブジェクトをリクエストします。
4. S3 はオブジェクトを CloudFront に返します。
5. CloudFront はオブジェクトをキャッシュします。
6. オブジェクトがビューワーに返されます。同じ CloudFront エッジロケーションに送信されるオブジェクトに対する後続のリクエストは、CloudFront キャッシュから処理されます。

ソリューションをデプロイする

この安全な静的ウェブサイトソリューションをデプロイするには、次のいずれかのオプションを選択できます。

- AWS CloudFormation コンソールを使用してデフォルトコンテンツでソリューションをデプロイしてから、ウェブサイトのコンテンツを Amazon S3 にアップロードします。
- ソリューションをコンピュータにクローンして、ウェブサイトのコンテンツを追加します。次に、AWS Command Line Interface (AWS CLI) を使用してソリューションをデプロイします。

> **Note**
>
> 米国東部 (バージニア北部) リージョンを使用して CloudFormation テンプレートをデプロイする必要があります。

トピック

前提条件

このソリューションを使用するには、次の前提条件が必要です。

- Amazon Route 53 ホストゾーンを指している登録済みドメイン名 (example.com など)。ホストゾーンは、このソリューションのデプロイ先と同じ AWS アカウントにある必要があります。登録済みドメイン名がない場合、Route 53 で登録できます。登録済みドメイン名を持っていても、Route 53 のホストゾーンをポイントしていないという場合は、Route 53 を DNS サービスとして設定します。
- IAM ロールを作成する CloudFormation テンプレートを起動するための AWS Identity and Access Management (IAM) 許可、およびソリューション内のすべての AWS のリソースを作成するための許可。

このソリューションの使用中に発生したコストは、お客様の負担となります。コストの詳細については、各 AWS のサービスの料金ページを参照してください。

AWS CloudFormation コンソールを使用する

CloudFormation コンソールを使用してデプロイするには

1. [Launch on AWS] (Launch で起動) を選択して、AWS CloudFormation コンソールでこのソリューションを開きます。必要に応じて、AWS アカウントにサインインします。

2. CloudFormation コンソールで [スタックを作成] ウィザードが開きます。このソリューションの CloudFormation テンプレートを指定するフィールドが事前に入力されています。

   ページの最下部にある [Next] を選択します。

3. [スタック詳細の指定] ページで、次のフィールドに値を入力します。

   - SubDomain - ウェブサイトで使用するサブドメインを入力します。たとえば、サブドメインが www の場合、ウェブサイトは *www*.example.com で利用できます。(次の箇条書きで説明するように、example.com をドメイン名に置き換えます)。
   - DomainName - *example.com* などのドメイン名を入力します。このドメインは Route 53 ホストゾーンを指している必要があります。
   - HostedZoneId — ドメイン名の Route 53 ホストゾーン ID。
   - CreateApex – (オプション) CloudFront 設定にドメイン頂点 (example.com) へのエイリアスを作成します。

4. 完了したら、[次へ] を選択します。
5. (オプション) [スタックオプションの設定] ページで、タグおよびその他のスタックオプションを追加します。
6. 完了したら、[次へ] を選択します。
7. [レビュー] ページで、ページの下部までスクロールし、[機能] セクションの 2 つのボックスを選択します。これらの機能により、CloudFormation はスタックのリソースへのアクセスを許可する IAM ロールを作成し、リソースに動的に名前を付けることができます。
8. [スタックの作成] を選択します。
9. スタックの作成が完了するまで待ちます。スタックはネストされたスタックを作成します。完了までに数分かかることがあります。完了すると、[ステータス] が [CREATE_COMPLETE] に変わります。

ステータスが [CREATE_COMPLETE] の場合、https://*www.example.com* に移動してウェブサイトを表示します (www.example.com は、ステップ 3 で指定したサブドメイン名およびドメイン名に置き換えます)。ウェブサイトのデフォルトコンテンツが表示されます。

ウェブサイトのデフォルトのコンテンツを独自のコンテンツに置き換えるには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. 名前が amazon-cloudfront-secure-static-site-s3bucketroot- で始まるバケットを選択します。

> **Note**
>
> s3bucketlogs ではなく、名前に s3bucketroot を含むバケットを選択してください。名前に s3bucketroot が含まれるバケットには、ウェブサイトのコンテンツが含まれています。s3bucketlogs が含まれるバケットには、ログファイルのみ含まれています。

3. ウェブサイトのデフォルトコンテンツを削除し、独自のコンテンツをアップロードします。

> **Note**
>
> このソリューションのデフォルトコンテンツでウェブサイトを表示した場合、デフォルトコンテンツの一部が CloudFront エッジロケーションにキャッシュされている可能性があります。更新されたウェブサイトのコンテンツがビューワーに確実に表示されるようにするには、ファイルを無効にして、CloudFront エッジロケーションからキャッシュされたコピーを削除します。詳細については、「ファイルを無効化してコンテンツを削除する」を参照してください。

ソリューションのクローンをローカルで作成する

前提条件

このソリューションをデプロイする前にウェブサイトのコンテンツを追加するには、ソリューションのアーティファクトをローカルでパッケージ化する必要があります。これには、Node.js と npm が必要です。詳細については、「https://www.npmjs.com/get-npm」を参照してください。

ウェブサイトのコンテンツを追加し、ソリューションをデプロイするには

1. からソリューションをクローンまたはダウンロードしますhttps://github.com/aws-samples/amazon-cloudfront-secure-static-site クローンを作成またはダウンロードしたら、コマンドプロンプトまたはターミナルを開き、`amazon-cloudfront-secure-static-site` フォルダに移動します。
2. 次のコマンドを実行し、ソリューションのアーティファクトをインストールしてパッケージ化します。

   ```
   make package-static
   ```

3. ウェブサイトのコンテンツを `www` フォルダにコピーし、デフォルト Web サイトのコンテンツを上書きします。
4. 次の AWS CLI コマンドを実行して、ソリューションのアーティファクトを保存する Amazon S3 バケットを作成します。*`example-bucket-for-artifacts`* を独自のバケット名に置き換えます。

   ```
   aws s3 mb s3://example-bucket-for-artifacts --region us-east-1
   ```

5. 次の AWS CLI コマンドを実行し、ソリューションのアーティファクトを CloudFormation テンプレートとしてパッケージ化します。*`example-bucket-for-artifacts`* を、前のステップで作成したバケットの名前に置き換えます。

   ```
   aws cloudformation package \
       --region us-east-1 \
       --template-file templates/main.yaml \
       --s3-bucket example-bucket-for-artifacts \
       --output-template-file packaged.template
   ```

6. 次のコマンドを実行し、CloudFormation を使用してソリューションをデプロイし、次の値を置き換えます。

   - *`your-CloudFormation-stack-name`* - CloudFormation スタックの名前で置き換えます。
   - *`example.com`* - ドメイン名で置き換えます。このドメインは、同じ AWS アカウント の Route 53 ホストゾーンをポイントしている必要があります。

- *www* - ウェブサイトで使用するサブドメインに置き換えます。たとえば、サブドメインが www の場合、ウェブサイトは www.example.com で利用できます。
- *hosted-zone-ID* — ドメイン名の Route 53 ホストゾーン ID に置き換えます。

```
aws cloudformation deploy \
    --region us-east-1 \
    --stack-name your-CloudFormation-stack-name \
    --template-file packaged.template \
    --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND \
    --parameter-overrides DomainName=example.com SubDomain=www HostedZoneId=hosted-zone-ID
```

- (オプション) ドメイン頂点を使用してスタックをデプロイするには、代わりに次のコマンドを実行します。

```
aws --region us-east-1 cloudformation deploy \
    --stack-name your-CloudFormation-stack-name \
    --template-file packaged.template \
    --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND \
    --parameter-overrides  DomainName=example.com SubDomain=www
 HostedZoneId=hosted-zone-ID CreateApex=yes
```

7. CloudFormation スタックの作成が完了するまで待ちます。スタックはネストされたスタックを作成します。完了までに数分かかることがあります。完了すると、[ステータス] が [CREATE_COMPLETE] に変わります。

   ステータスが [CREATE_COMPLETE] に変わったら、https://www.example.com に移動してウェブサイトを表示します (www.example.com は、前のステップで指定したサブドメイン名およびドメイン名に置き換えます)。ウェブサイトのコンテンツが表示されます。

アクセスログの検索

このソリューションでは、CloudFront ディストリビューションのアクセスログが有効になります。ディストリビューションのアクセスログを見つけるには、以下のステップを実行します。

ディストリビューションのアクセスログを見つけるには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。

2. 名前が amazon-cloudfront-secure-static-site-s3bucketlogs- で始まるバケットを選択します。

> **Note**
>
> s3bucketroot ではなく、名前に s3bucketlogs を含むバケットを選択してください。名前に s3bucketlogs が含まれるバケットには、ログファイルが含まれています。s3bucketroot が含まれるバケットには、ウェブサイトのコンテンツが含まれています。

3. cdn という名前のフォルダには、CloudFront アクセスログが含まれています。

ディストリビューションの設定

Amazon CloudFront ディストリビューションを作成して、コンテンツを配信する場所と、コンテンツ配信の追跡と管理の方法の詳細を CloudFront に指示します。

次の構成設定から選択します。

- コンテンツオリジン — CloudFront が配信するファイルの取得元である Amazon S3 バケット、AWS Elemental MediaPackage チャネル、AWS Elemental MediaStore コンテナ、Elastic Load Balancing ロードバランサー、または HTTP サーバーです。1 つのディストリビューションで、最大で 25 のオリジンの任意に組み合わせて指定できます。
- アクセス - ファイルをすべてのユーザーが使用できるようにするか、または一部のユーザーにアクセスを制限するか。
- セキュリティ - AWS WAF 保護を有効にして、HTTPS を使用したコンテンツへのアクセスを必須にするかどうか。
- キャッシュキー - キャッシュキーに含める値 (存在する場合)。キャッシュキーは、特定のディストリビューションのキャッシュ内の各ファイルを一意に識別します。
- オリジンリクエスト設定 - CloudFront でオリジンに送信するリクエストに HTTP ヘッダー、Cookie、またはクエリ文字列を含めるかどうか。
- 地理的制限 — CloudFront で特定の国のユーザーがコンテンツにアクセスできないようにするかどうか。
- ログ — CloudFront で標準ログを作成するか、ビューワーのアクティビティを示すリアルタイムログを作成するかどうか。

詳細については、「ディストリビューション設定リファレンス」を参照してください。

AWS アカウントごとに作成できるディストリビューションの現在の最大数については、「ディストリビューションの一般的なクォータ」を参照してください。ディストリビューションごとにサービスが可能なファイルの最大数はありません。

ディストリビューションを使用して、HTTP または HTTPS 経由で以下のコンテンツを供給できます。

- HTTP または HTTPS を使用した静的および動的なコンテンツ (HTML、CSS、JavaScript、イメージファイルなど) のダウンロード。

- Apple HTTP Live Streaming (HLS) や Microsoft Smooth Streaming など、さまざまな形式のビデオオンデマンド。詳細については、「CloudFront でビデオオンデマンドを配信する」を参照してください。
- ライブイベント。リアルタイムのミーティング、会議、コンサートなど。ライブストリーミングの場合は、AWS CloudFormation スタックを使用して自動的にディストリビューションを作成できます。詳細については、「CloudFront と AWS Media Services でライブストリーミングビデオを配信する」を参照してください。

以下のトピックでは、CloudFront ディストリビューションの詳細と、ビジネスニーズに合わせてディストリビューションを設定する方法について説明します。ディストリビューションの作成については、「ディストリビューションを作成する」を参照してください。

トピック

ディストリビューションを作成する

このトピックでは、CloudFront コンソールを使用してディストリビューションを作成する方法について説明します。

ディストリビューションの作成の概要

1. 1 つ以上の Amazon S3 バケットを作成するか、HTTP サーバーをオリジンサーバーとして構成します。オリジンとは、コンテンツのオリジナルバージョンを保存する場所です。CloudFront は、ファイルに対するリクエストを受け取ると、オリジンにアクセスし、エッジロケーション

に配信するファイルを取得します。オリジンサーバーとして、10 個の Amazon S3 バケットと HTTP サーバーの任意の組み合わせを使用できます。

- Amazon S3 を使用する場合は、バケット名をすべて小文字にする必要があります。また、バケット名にスペースを含めることはできません。
- Amazon EC2 サーバーまたは別のカスタムオリジンを使用する場合は、「Amazon EC2 (または別のカスタムオリジン) を使用する」を確認してください。
- ディストリビューションに対して作成できるオリジンの現在の最大数について、またはクォータの引き上げを要求するには、「ディストリビューションの一般的なクォータ」を参照してください。

2. コンテンツをオリジンサーバーにアップロードします。オブジェクトをパブリックに読み出し可能にするか、または CloudFront の署名付き URL を使用して、コンテンツへのアクセスを制限できます。

> **Important**
>
> オリジンサーバーのセキュリティを確保する責任はお客様にあります。サーバーへのアクセス権限を CloudFront に持たせる必要があります。また、コンテンツを保護するためのセキュリティ設定が必要があります。

3. CloudFront ディストリビューションの作成

   - CloudFront コンソールでディストリビューションを作成する詳細な手順については、「ディストリビューションを作成する」を参照してください。
   - CloudFront API を使用してディストリビューションを作成する方法については、「Amazon CloudFront API リファレンス」の「CreateDistribution」を参照してください。

4. (オプション) CloudFront コンソールを使用してディストリビューションを作成する場合は、ディストリビューションのキャッシュ動作またはオリジンをさらに作成します。動作およびオリジンの詳細については、「CloudFront ディストリビューションを更新するには」を参照してください。

5. ディストリビューションをテストします。テストの詳細については、「ディストリビューションのテスト」を参照してください。

6. ステップ 3 でディストリビューションを作成した後に CloudFront から返されたドメイン名を使用して、お客様のコンテンツにアクセスするためのウェブサイトまたはアプリケーションを開発します。例えば、CloudFront がディストリビューションのドメイン名として d111111abcdef8.cloudfront.net を返した場合、Amazon S3 バケット内または

HTTP サーバーのルートディレクトリ内のファイル image.jpg の URL は https://d111111abcdef8.cloudfront.net/image.jpg になります。

ディストリビューションの作成時に 1 つ以上の代替ドメイン名 (CNAME) を指定した場合、独自のドメイン名を使用できます。この場合、image.jpg の URL を https://www.example.com/image.jpg にすることができます。

次の点に注意してください。

- 署名付き URL を使用してコンテンツへのアクセスを制限する場合は、「署名付き URL と署名付き Cookie を使用したプライベートコンテンツを提供する」を参照してください。
- 圧縮されたコンテンツを供給する場合は、「圧縮ファイルを供給する」を参照してください。
- Amazon S3 オリジンおよびカスタムオリジンに対する CloudFront のリクエスト動作およびレスポンス動作については、「リクエストとレスポンスの動作」を参照してください。

トピック

コンソールに CloudFront ディストリビューションを作成する

ディストリビューションを作成するには (コンソール)

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで [ディストリビューション] を選択し、[ディストリビューションを作成] を選択します。
3. ディストリビューションの設定項目を指定します。詳細については、「ディストリビューション設定リファレンス」を参照してください。
4. 変更を保存します。
5. CloudFront でディストリビューションを作成すると、ディストリビューションの [ステータス] 列の値が、[デプロイ中] からディストリビューションをデプロイした日時に変わります。ディストリビューションを有効にすることを選択していた場合、この時点でリクエストを処理する準備が整います。

CloudFront によってディストリビューションに割り当てられたドメイン名がディストリビューションのリストに表示されます。(ドメイン名は、選択されたディストリビューションの [General] タブにも表示されます)。

> **Tip**
>
> 「代替ドメイン名 (CNAME) を追加することによって、カスタム URL を使用する」の手順に従って、CloudFront によって割り当てられた名前の代わりに、代替ドメイン名を使用できます。

6. ディストリビューションをデプロイする場合は、新しい CloudFront URL または CNAME を使用してコンテンツにアクセスできることを確認します。詳細については、「ディストリビューションのテスト」を参照してください。

CloudFront でコンソールに表示される値

新しい ディストリビューションの作成や、既存のディストリビューションの更新を行う場合、CloudFront は以下の情報を CloudFront コンソールに表示します。

> **Note**
>
> 有効な信頼済み署名者、つまり有効な CloudFront キーペアを持つ AWS アカウントのうち有効な署名付き URL の作成に使用できるものは、CloudFront コンソールに現在表示されません。

ディストリビューション ID

CloudFront API を使用してディストリビューションに対するアクションを実行する場合、ディストリビューション ID を使用して、どのディストリビューションを使用するかを指定します (例: `EDFDVBD6EXAMPLE`)。ディストリビューションのディストリビューション ID を変更することはできません。

デプロイとステータス

ディストリビューションをデプロイすると、[最終変更日] 列に [デプロイ中] ステータスが表示されます。ディストリビューションのデプロイが完了するのを待ち、[ステータス] 列に [有効] と表示されるのを確認します。詳細については、「ディストリビューションの状態」を参照してください。

最終更新日時

ディストリビューションが最後に変更された日時。ISO 8601 形式が使用されます (例: 2012-05-19T19:37:58Z)。詳しくは、「https://www.w3.org/TR/NOTE-datetime」を参照してください。

ドメイン名

オブジェクトへのリンク内で、ディストリビューションのドメイン名を使用します。たとえば、ディストリビューションのドメイン名が `d111111abcdef8.cloudfront.net` の場合、`/images/image.jpg` へのリンクは `https://d111111abcdef8.cloudfront.net/images/image.jpg` になります。ディストリビューションの CloudFront ドメイン名を変更することはできません。オブジェクトへのリンクの CloudFront URL の詳細については、「CloudFront でファイルの URL 形式をカスタマイズする」を参照してください。

1 つ以上の代替ドメイン名 (CNAME) を指定した場合、オブジェクトへのリンクに、CloudFront ドメイン名ではなく独自のドメイン名を使用できます。CNAME の詳細については、「代替ドメイン名 (CNAME)」を参照してください。

> **Note**
>
> CloudFront ドメイン名は一意です。ディストリビューションのドメイン名は以前のディストリビューションで使用されておらず、今後の別のディストリビューションでも再利用されません。

その他のリンク

ディストリビューションの作成の詳細については、以下のリンクを参照してください。

- オリジンアクセスコントロール (OAC) で Amazon Simple Storage Service (Amazon S3) バケットオリジンを使用するディストリビューションを作成する方法については、「基本的な CloudFront ディストリビューションの開始方法」を参照してください。
- CloudFront API を使用してディストリビューションを作成する方法については、「Amazon CloudFront API リファレンス」の「CreateDistribution」を参照してください。
- ディストリビューションの更新については (例えば、キャッシュ動作を追加または変更するため)、「ディストリビューションを更新する」を参照してください。

- AWS アカウントごとに作成できるディストリビューションの現在の最大数、またはクォータの引き上げを要求するには、「ディストリビューションの一般的なクォータ」を参照してください。

ディストリビューション設定リファレンス

CloudFront コンソールを使用して新しいディストリビューションの作成や、既存のディストリビューションの更新を行う場合、以下の値を指定します。

CloudFront コンソールを使用してディストリビューションを作成または更新する方法の詳細については、「the section called “ディストリビューションを作成する”」または「the section called “ディストリビューションを更新する”」を参照してください。

トピック

- オリジンの設定
- キャッシュ動作の設定
- ディストリビューションの設定
- カスタムエラーページとエラーキャッシュ
- 地理的制限

オリジンの設定

CloudFront コンソールを使用してディストリビューションを作成または更新する場合、1 つ以上のロケーションに関する情報を指定します。これは、オリジンと呼ばれるものであり、ウェブコンテンツのオリジナルバージョンを保存する場所です。CloudFront はオリジンからウェブコンテンツを取得し、世界中のエッジサーバーネットワークを経由してビューワーにウェブコンテンツを供給します。

ディストリビューションに対して作成できるオリジンの現在の最大数について、またはクォータの引き上げを要求するには、「the section called “ディストリビューションの一般的なクォータ”」を参照してください。

オリジンを削除する場合は、まず、そのオリジンに関連付けられたキャッシュ動作を編集または削除する必要があります。

> **Important**
>
> オリジンを削除する場合は、そのオリジンによって以前供給されていたファイルが別のオリジンで利用可能であり、現在、そのファイルへのリクエストがキャッシュ動作によって新しいオリジンにルーティングされていることを確認します。

ディストリビューションを作成または更新する場合、オリジンごとに以下の値を指定します。

トピック

オリジンドメイン

オリジンドメインは、CloudFront がこのオリジンのオブジェクトの取得先としている Amazon S3 バケットまたは HTTP サーバーの DNS ドメイン名です。以下に例を示します。

- Amazon S3 バケット – *DOC-EXAMPLE-BUCKET*.s3.*us-west-2*.amazonaws.com

> **Note**
>
> S3 バケットを最近作成した場合、CloudFront ディストリビューションは最大 24 時間以内に HTTP 307 Temporary Redirect レスポンスを返す可能性があります。S3 バケット名がすべての AWS リージョンに反映されるまでに、最大 24 時間かかることがあります。

反映が完了すると、ディストリビューションは自動的にこれらのリダイレクトレスポンスの送信を停止します。何もする必要はありません。詳細については、「Why am I getting an HTTP 307 Temporary Redirect response from Amazon S3?」と「一時的なリクエストのリダイレクト」を参照してください。

- ウェブサイトとして設定された Amazon S3 バケット – *DOC-EXAMPLE-BUCKET*.s3-website.*us-west-2*.amazonaws.com
- MediaStore コンテナ – *examplemediastore*.data.mediastore.*us-west-1*.amazonaws.com
- MediaPackage エンドポイント – *examplemediapackage*.mediapackage.*us-west-1*.amazonaws.com
- Amazon EC2 インスタンス – *ec2-203-0-113-25*.compute-1.amazonaws.com
- Elastic Load Balancing ロードバランサー – *example-load-balancer-1234567890*.*us-west-2*.elb.amazonaws.com
- 独自のウェブサーバー – https://www.example.com

[Origin Domain Name] (オリジンドメイン名) フィールドでドメイン名を選択するか、名前を入力します。ドメイン名では、大文字と小文字が区別されません。

オリジンが Amazon S3 バケットの場合は、次の点に注意してください。

- バケットがウェブサイトとして構成されている場合は、バケットの Amazon S3 静的ウェブサイトホスティングエンドポイントを入力します。[Origin domain] (オリジンドメイン) フィールドのバケット一覧からバケット名を選択しないでください。静的ウェブサイトホスティングエンドポイントは、Amazon S3 コンソールの [Static website hosting] (静的ウェブサイトホスティング) の [Properties] (プロパティ) ページに表示されます。詳細については、「the section called “ウェブサイトのエンドポイントとして設定された Amazon S3 バケットを使用する”」を参照してください。
- バケットに Amazon S3 Transfer Acceleration を設定した場合、[Origin domain] (オリジンドメイン) に s3-accelerate エンドポイントを指定しないでください。
- 使用しているバケットが別の AWS アカウントのものであり、そのバケットがウェブサイトとして構成されていない場合は、次の形式で名前を入力します。

  bucket-name.s3.*region*.amazonaws.com

バケットが米国リージョンにあり、Amazon S3 がバージニア北部の施設にリクエストをルーティングするように設定する場合は、次の形式を使用します。

bucket-name.s3.us-east-1.amazonaws.com

- CloudFront のオリジンアクセスコントロールを使用して Amazon S3 内のコンテンツを保護しない限り、ファイルをパブリックに読み取り可能とする必要があります。アクセスコントロールの詳細については、「the section called “Amazon Simple Storage Service オリジンへのアクセスを制限する”」を参照してください。

> **Important**
>
> オリジンが Amazon S3 バケットの場合、バケット名は DNS 命名要件に準拠する必要があります。詳細については、Amazon Simple Storage Service ユーザーガイドのバケットの制約と制限を参照してください。

オリジンの [Origin domain] (オリジンドメイン) の値を変更すると、CloudFront は CloudFront エッジロケーションに対する変更の複製を直ちに開始します。該当するエッジロケーションでディストリビューション構成が更新されるまで、CloudFront はリクエストを以前のオリジンに引き続き転送します。該当のエッジロケーションでディストリビューション構成が更新されると、CloudFront は新しいオリジンへのリクエストの転送を直ちに開始します。

オリジンを変更しても、CloudFront が、新しいオリジンからのオブジェクトでエッジキャッシュを再生成する必要はありません。アプリケーション内でビューワーのリクエストが変更されていない限り、各オブジェクトの TTL の有効期限が切れるか、要求頻度の低いオブジェクトが削除されるまで、CloudFront は、引き続き、エッジキャッシュに既に保持されているオブジェクトを供給します。

プロトコル (カスタムオリジンのみ)

> **Note**
>
> これは、カスタムオリジンにのみ適用されます。

CloudFront がオリジンからオブジェクトをフェッチするときに使用するプロトコルポリシー。

次のいずれかの値を選択します。

- [HTTP only] (HTTP のみ): CloudFront は HTTP のみを使用してオリジンにアクセスします。

> **Important**
>
> Amazon S3 は静的ウェブサイトホスティングエンドポイントをサポートしていないため、オリジンが Amazon S3 の静的ウェブサイトホスティングエンドポイントの場合、[HTTPS only] (HTTP のみ) がデフォルト設定です。CloudFront コンソールは、エンドポイントをホストする Amazon S3 の静的ウェブサイトのこの設定の変更をサポートしていません。

- [HTTPS only] (HTTPS のみ): CloudFront は HTTPS のみを使用してオリジンにアクセスします。
- [Match viewer] (ビューワーに合わせる): CloudFront は、ビューワーのリクエストのプロトコルに応じて HTTP または HTTPS を使用し、オリジンと通信します。ビューワーが HTTP と HTTPS の両方のプロトコルを使用してリクエストを行った場合も、CloudFront がオブジェクトをキャッシュするのは 1 回だけです。

> **Important**
>
> CloudFront がこのオリジンに転送する HTTPS ビューワーリクエストの場合、オリジンサーバーの SSL 証明書のドメイン名のうち 1 つは、[Origin domain] (オリジンドメイン) で指定したドメイン名と一致する必要があります。一致しない場合、CloudFront は、ビューワーリクエストに対して、リクエストされたオブジェクトではなく、HTTP ステータスコード 502 (不正なゲートウェイ) を返します。詳細については、「the section called “CloudFront で SSL/TLS 証明書を使用するための要件”」を参照してください。

トピック

- HTTP ポート
- HTTPS ポート
- 最小限のオリジン SSL プロトコル

HTTP ポート

> **Note**
>
> これは、カスタムオリジンにのみ適用されます。

(オプション) カスタムオリジンがリスンする HTTP ポートを指定できます。有効な値には、ポート 80、443、および 1024～65535 が含まれます。デフォルト値はポート 80 です。

> **Important**
>
> オリジンが Amazon S3 の静的ウェブサイトホスティングエンドポイントの場合、ポート 80 がデフォルト設定です。Amazon S3 は、静的ウェブサイトホスティングエンドポイントではポート 80 のみサポートするためです。CloudFront コンソールは、エンドポイントをホストする Amazon S3 の静的ウェブサイトのこの設定の変更をサポートしていません。

HTTPS ポート

> **Note**
>
> これは、カスタムオリジンにのみ適用されます。

(オプション) カスタムオリジンがリスンする HTTPS ポートを指定できます。有効な値には、ポート 80、443、および 1024～65535 が含まれます。デフォルト値はポート 443 です。[Protocol] (プロトコル) は、[HTTP only] (HTTP のみ) に設定されます。[HTTPS port] (HTTPS ポート) の値を指定することはできません。

最小限のオリジン SSL プロトコル

> **Note**
>
> これは、カスタムオリジンにのみ適用されます。

オリジンに HTTPS 接続を設立する場合には、CloudFront が使用できる最小限の TLS/SSL プロトコルを選択します。より低い TLS プロトコルは安全性が低いため、オリジンがサポートする最新の TLS を使用することが推奨されます。[Protocol] (プロトコル) は、[HTTP only] (HTTP のみ) に設定されます。[Minimum origin SSL protocol] (最小限のオリジン SSL プロトコル) の値を指定することはできません。

CloudFront が使用する TLS/SSL プロトコルを CloudFront API を使用して設定する場合は、最小プロトコルを設定することはできません。代わりに、CloudFront がオリジンで使用できるすべて

の TLS/SSL プロトコルを指定します。詳細については、Amazon CloudFront API リファレンスの「OriginSslProtocols」を参照してください。

オリジンのパス

CloudFront がオリジンのディレクトリにコンテンツがリクエストされるようにするには、スラッシュ (/) 以降のディレクトリパスを入力します。CloudFront は、[Origin domain] (オリジンドメイン) の値にディレクトリ名を追加します。例えば、**`cf-origin.example.com/production/images`** などです。パスの末尾にはスラッシュ (/) を付けないでください。

例えば、特定のディストリビューションに対して次の値を指定したとします。

- [Origin domain] (オリジンドメイン) - **`DOC-EXAMPLE-BUCKET`** と名前の付いた Amazon S3 バケット
- オリジンのパス - **`/production`**
- 代替ドメイン名 (CNAME) – **`example.com`**

ユーザーがブラウザに `example.com/index.html` と入力すると、CloudFront が `DOC-EXAMPLE-BUCKET/production/index.html` のリクエストを Amazon S3 に送信します。

ユーザーがブラウザに `example.com/acme/index.html` と入力すると、CloudFront が `DOC-EXAMPLE-BUCKET/production/acme/index.html` のリクエストを Amazon S3 に送信します。

名前

名前は、このディストリビューション内でこのオリジンを一意に識別する文字列です。デフォルトのキャッシュ動作に加えてキャッシュ動作を作成する場合、ここで指定した名前を使用して、そのキャッシュ動作のパスパターンにリクエストが一致した場合に CloudFront がリクエストをルーティングするオリジンを識別します。

オリジンアクセス (Amazon S3 オリジンのみ)

> **Note**
>
> これは、Amazon S3 バケットオリジン (S3 静的ウェブサイトエンドポイントを使用していないオリジン) にのみ適用されます。

Amazon S3 バケットオリジンへのアクセスを特定の CloudFront ディストリビューションのみに制限できるようにする場合は、[オリジンアクセスコントロール設定 (推奨)] を選択します。

Amazon S3 バケットオリジンがパブリックにアクセス可能な場合は、[パブリック] を選択します。

詳しくは、「the section called “Amazon Simple Storage Service オリジンへのアクセスを制限する”」を参照してください。

CloudFront URL のみ使用してカスタムオリジン内のオブジェクトにアクセスするよう要求する方法の詳細については、「the section called “カスタムオリジンのファイルへのアクセスを制限する”」を参照してください。

カスタムヘッダーを追加する

CloudFront がリクエストをオリジンに転送するごとにカスタムヘッダーを追加するようにする場合は、ヘッダー名とその値を指定します。詳細については、「the section called “オリジンリクエストにカスタムヘッダーを追加する”」を参照してください。

現在、追加できるカスタムヘッダーの最大数、カスタムヘッダー名と値の最大長、すべてのヘッダー名と値の全長の最大長については、クォータ を参照してください。

Origin Shield を有効にする

[Yes] (はい) を選択して、CloudFront の Origin Shield を有効にします。Origin Shield の詳細については、the section called “Origin Shield の使用” を参照してください。

接続の試行

CloudFront がオリジンへの接続を試行する回数を設定できます。試行回数として 1、2、または 3 を指定できます。デフォルト値 (特に指定しない場合) は 3 です。

この設定を [Connection timeout] (接続タイムアウト) とともに使用すると、セカンダリオリジンに接続しようとしたり、ビューワーにエラーレスポンスを返したりするまでに CloudFront が待機する時間の長さを指定できます。デフォルトでは、CloudFront はセカンダリオリジンへの接続を試行したり、エラーレスポンスを返したりする前に 30 秒 (それぞれ 10 秒間の試行が 3 回) 待機します。試行回数を減らすか、接続タイムアウトを短くするか、その両方を行うことで、この時間を短縮できます。

接続試行が指定した回数失敗した場合、CloudFront は次のいずれかを実行します。

- オリジンがオリジングループの一部である場合、CloudFront はセカンダリオリジンへの接続を試みます。セカンダリオリジンへの接続試行が指定された回数失敗した場合、CloudFront はビューワーにエラーレスポンスを返します。
- オリジンがオリジングループの一部でない場合、CloudFront はビューワーにエラーレスポンスを返します。

カスタムオリジン (静的ウェブサイトホスティングで設定された Amazon S3 バケットを含む) では、この設定は、CloudFront がオリジンからレスポンスを待機する回数も指定します。詳細については、「the section called “応答タイムアウト (カスタムオリジンのみ)”」を参照してください。

接続タイムアウト

接続タイムアウトは、オリジンへの接続を確立しようとしたときに CloudFront が待機する秒数です。1 ~ 10 (両端の値を含む) の秒数を指定できます。デフォルトのタイムアウト (特に指定しない場合) は 10 秒です。

この設定を [Connection attempts] (接続の試行) とともに使用すると、セカンダリオリジンに接続しようとしたり、ビューワーにエラーレスポンスを返したりするまでに CloudFront が待機する時間の長さを指定できます。デフォルトでは、CloudFront はセカンダリオリジンへの接続を試行したり、エラーレスポンスを返したりする前に 30 秒 (それぞれ 10 秒間の試行が 3 回) 待機します。試行回数を減らすか、接続タイムアウトを短くするか、その両方を行うことで、この時間を短縮できます。

CloudFront が指定した秒数以内にオリジンへの接続を確立しない場合、CloudFront は次のいずれかを実行します。

- 指定した [Connection attempts] (接続の試行) が 1 を超える場合、CloudFront は接続の確立を再試行します。CloudFront は、[Connection attempts] (接続の試行) の値に従って最大 3 回試行します。
- すべての接続試行が失敗した場合で、オリジンがオリジングループの一部である場合、CloudFront はセカンダリオリジンへの接続を試みます。セカンダリオリジンへの接続試行が指定された回数失敗した場合、CloudFront はビューワーにエラーレスポンスを返します。
- すべての接続試行が失敗した場合で、オリジンがオリジングループの一部でない場合、CloudFront はエラーレスポンスをビューワーに返します。

応答タイムアウト (カスタムオリジンのみ)

オリジン応答タイムアウト (オリジンの読み取りタイムアウトまたはオリジンリクエストタイムアウトとも呼ばれる) は、次の両方の値に適用されます。

- CloudFront がリクエストをオリジンに転送してからレスポンスを受け取るまでの待機時間 (秒)
- CloudFront がオリジンからレスポンスのパケットを受け取ってから次のパケットを受け取るまでの待機時間 (秒)

> **Tip**
>
> ビューワーで HTTP 504 ステータスコードエラーが発生しているために、タイムアウト値を引き上げる必要がある場合は、タイムアウト値を変更する前に、それらのエラーを回避するその他の方法を検討します。「the section called “HTTP 504 ステータスコード (Gateway Timeout)”」でトラブルシューティングのヒントを参照してください。

CloudFront の動作は、ビューワーリクエストの HTTP メソッドによって決まります。

- `GET` および `HEAD` リクエスト – 応答タイムアウトの期間内にオリジンが応答しない場合、または応答を停止した場合、CloudFront は接続を中断します。CloudFront は the section called “接続の試行” の値に従って接続を再試行します。
- `DELETE`、`OPTIONS`、`PATCH`、`PUT`、`POST` の各リクエスト – オリジンが読み取りタイムアウトの期間中に応答しない場合、CloudFront は接続を中断し、オリジンへの接続を再試行しません。クライアントは、必要に応じてリクエストを再送信できます。

キープアライブタイムアウト (カスタムオリジンのみ)

キープアライブタイムアウトは、CloudFront がレスポンスの最後のパケットを取得した後にカスタムオリジンへの接続を維持する時間 (秒) です。持続的接続を維持すると、TCP 接続の再構築に必要な時間と後続のリクエストに対する別の TLS ハンドシェイクの実行に必要な時間を節約できます。キープアライブタイムアウトを増やすと、ディストリビューションの接続あたりのリクエストメトリクスの改善に役立ちます。

> **Note**
>
> [Origin Keep-alive Timeout] (オリジンキープアライブタイムアウト) の値が有効になるためには、永続接続を許可するようにオリジンが設定されている必要があります。

レスポンスとキープアライブのタイムアウトクォータ

> **Note**
>
> これは、カスタムオリジンにのみ適用されます。

- レスポンスタイムアウトのデフォルトは 30 秒です。
- キープアライブタイムアウトのデフォルトは 5 秒です。
- どちらのクォータでも、1～60 秒の値を指定できます。増加をリクエストするには、AWS Support Center Console/ でケースを作成します。

AWS アカウント についてタイムアウトの増加をリクエストした後、ディストリビューションオリジンを更新して、必要なレスポンスタイムアウト値とキープアライブタイムアウト値になるようにします。アカウントのクォータを増やしても、オリジンは自動的に更新されません。例えば、Lambda@Edge 関数を使用してキープアライブタイムアウトを 90 秒に設定する場合、オリジンにはすでに 90 秒以上のキープアライブタイムアウトが必要です。そうしないと、Lambda@Edge 関数の実行に失敗することがあります。

ディストリビューションクォータの詳細については、「ディストリビューションの一般的なクォータ」を参照してください。

キャッシュ動作の設定

キャッシュ動作を設定すると、ウェブサイトにあるファイルの特定の URL パスパターンに応じてさまざまな CloudFront 機能を設定できます。たとえば、CloudFront のオリジンサーバーとして使用しているウェブサーバーの `.jpg` ディレクトリ内にあるすべての `images` ファイルに 1 つのキャッシュ動作を適用することができます。キャッシュ動作ごとに構成可能な機能には以下のようなものがあります。

- パスパターン
- CloudFront ディストリビューションに対して複数のオリジンを構成した場合、CloudFront でリクエストを転送するオリジン。
- クエリ文字列をオリジンに転送するかどうか
- 指定したファイルへのアクセスに署名付き URL を必要とするかどうか
- これらのファイルへのアクセスに HTTPS を使用するようユーザーに要求するかどうか

- オリジンがファイルに追加する Cache-Control ヘッダーの値に関係なく、これらのファイルを CloudFront キャッシュに保持する最小時間

新しいディストリビューションを作成する場合、デフォルトのキャッシュ動作の設定を指定します。デフォルトのキャッシュ動作では、ディストリビューションの作成時に指定されたオリジンにすべてのリクエストが自動的に転送されます。ディストリビューションを作成した後、追加のキャッシュ動作を作成し、パスパターン (例: *.jpg) に一致するオブジェクトのリクエストを受け取ったときに CloudFront がどのように応答するかを定義できます。追加のキャッシュ動作を定義した場合、デフォルトのキャッシュ動作は常に最後に処理されます。他のキャッシュ動作は、CloudFront コンソールに表示された順序で処理されるか、CloudFront API が使用されている場合は、ディストリビューションの DistributionConfig エレメントに示された順序で処理されます。詳細については、「パスパターン」を参照してください。

キャッシュ動作を作成するときに、CloudFront がオブジェクトの取得先とするオリジンを 1 つ指定します。結果として、CloudFront がすべてのオリジンからオブジェクトを配信する場合、少なくともオリジンと同じ数のキャッシュ動作 (デフォルトのキャッシュ動作を含む) が必要です。2 つのオリジンとデフォルトのキャッシュ動作のみがある場合、デフォルトのキャッシュ動作によって、CloudFront は 1 つのオリジンからオブジェクトを取得します。その他のオリジンは一切使用されません。

ディストリビューションに対して作成できるキャッシュ動作の現在の最大数、またはクォータの引き上げを要求するキャッシュ動作の最大数については、「ディストリビューションの一般的なクォータ」を参照してください。

トピック

- パスパターン
- オリジンまたはオリジングループ
- ビューワープロトコルポリシー
- 許可される HTTP メソッド
- フィールドレベル暗号化の設定
- キャッシュされる HTTP メソッド
- 選択されたリクエストヘッダーに基づいたキャッシュ
- 許可リストヘッダー
- オブジェクトキャッシュ
- 最小 TTL

- 最大 TTL
- デフォルト TTL
- cookie の転送
- 許可リスト Cookie
- クエリ文字列の転送とキャッシュ
- クエリ文字列の許可リスト
- スムーズストリーミング
- ビューワーのアクセス制限 (署名付き URL または署名付き cookie の使用)
- 信頼された署名者
- AWS アカウント番号
- オブジェクトを自動的に圧縮する
- CloudFront イベント
- Lambda 関数の ARN
- 本文を含める

パスパターン

パスパターン (例: `images/*.jpg`) は、このキャッシュ動作をどのリクエストに割り当てるかを指定します。CloudFront がエンドユーザーリクエストを受け取ると、リクエストされたパスは、ディストリビューションに含まれるキャッシュ動作の順序でパスパターンと照合されます。最初の一致によって、そのリクエストに適用されるキャッシュ動作が決まります。たとえば、以下の 3 つのパスパターンを持つ 3 つのキャッシュ動作がこの順序で設定されているとします。

- `images/*.jpg`
- `images/*`
- `*.gif`

> **Note**
>
> オプションで、`/images/*.jpg` などのように、パスパターンの先頭にスラッシュ (/) を含めることができます。CloudFront の動作は先頭のスラッシュ (/) の有無で変化しません。パスの先頭に / を指定しなくても、この文字は自動的に暗黙で使用されます。CloudFront

は、先頭に / があってもなくても、パスを同じように扱います。例えば、CloudFront は /*product.jpg を *product.jpg と同じように扱います。

ファイル images/sample.gif のリクエストは 1 番目のパスパターンを満たさないため、関連付けられたキャッシュ動作はこのリクエストに適用されません。ファイルは 2 番目のパスパターンを満たします。リクエストは 3 番目のパスパターンにも一致しますが、2 番目のパスパターンに関連付けられたキャッシュ動作が適用されます。

Note

新しいディストリビューションを作成すると、デフォルトのキャッシュ動作の [Path Pattern (パスパターン)] の値は * (すべてのファイル) に設定され、この値は変更できません。この値によって、CloudFront は、オブジェクトに対するすべてのリクエストを、オリジンドメイン フィールドに指定されたオリジンに転送します。オブジェクトのリクエストが、他のどのキャッシュ動作のパスパターンにも一致しない場合、CloudFront は、デフォルトのキャッシュ動作に指定された動作を適用します。

Important

パスパターンとその順序を慎重に定義します。適切に定義されていない場合、コンテンツへの意図されないアクセスがユーザーに与えられる場合があります。たとえば、リクエストが、2 つのキャッシュ動作のパスパターンに一致したと仮定します。最初のキャッシュ動作は署名付き URL を要求しませんが、2 番目のキャッシュ動作は署名付き URL を要求します。ユーザーは署名付き URL を使用せずにオブジェクトにアクセスできます。これは、CloudFront が、最初の一致に関連付けられたキャッシュ動作を処理するためです。

MediaPackage チャネルを使用する場合は、オリジンのエンドポイントタイプに対して定義するキャッシュ動作に特定のパスパターンを含める必要があります。たとえば、DASH エンドポイントの場合は、[Path Pattern (パスパターン)] に「*.mpd」と入力します。詳細と具体的な手順については、「AWS Elemental MediaPackage でフォーマットされたライブ動画を配信する」を参照してください。

指定されたパスは、指定されたディレクトリ内のあらゆるファイル、および指定されたディレクトリ以下のサブディレクトリ内のあらゆるファイルのリクエストに適用されます。CloudFront はパ

スパターンを評価する際にクエリ文字列や Cookie を考慮しません。たとえば、images ディレクトリに product1 および product2 サブディレクトリが含まれる場合、パスパターン images/*.jpg は、images、images/product1、および images/product2 ディレクトリ内のあらゆる .jpg ファイルのリクエストに適用されます。異なるキャッシュ動作を、images/product1 および images ディレクトリのファイルではなく images/product2 ディレクトリのファイルに割り当てる場合、images/product1 用の独立したキャッシュ動作を作成し、そのキャッシュ動作を images ディレクトリ用のキャッシュ動作の上 (前) の位置に移動します。

パスパターンには、以下のワイルドカード文字を使用できます。

- * は、0 個以上の文字に一致します。
- ? は、正確に 1 個の文字に一致します。

以下の例を使用して、ワイルドカード文字がどのように機能するかを示します。

パスパターン	パスパターンに一致するファイル
*.jpg	すべての .jpg ファイル。
images/*.jpg	images ディレクトリ内、および images ディレクトリ下のサブディレクトリ内のすべての .jpg ファイル。
a*.jpg	• ファイル名が a で始まるすべての .jpg ファイル (例: apple.jpg 、appalachian_trail_2012_05_21.jpg)。 • ファイルパスが a で始まるすべての .jpg ファイル (例: abra/cadabra/magic.jpg)。
a??.jpg	ファイル名が a で始まり、ファイル名の後に正確に他の 2 文字が続くすべての .jpg ファイル (例: ant.jpg、abe.jpg)。
.doc	ファイル名拡張子が .doc で始まるすべてのファイル (例: .doc、.docx、.docm ファイル)。この場合、パスパターン *.doc? を使用

パスパターン	パスパターンに一致するファイル
	することはできません。このパスパターンは .doc ファイルのリクエストに適用されないためです。? ワイルドカード文字は正確に 1 個の文字を置き換えるものです。

パスパターンの最大長は 255 文字です。値には以下の文字を含めることができます。

- A~Z、a~z

  パスパターンでは大文字と小文字が区別されるので、パスパターン *.jpg はファイル LOGO.JPG には適用されません。
- 0-9
- _ - . * $ / ~ " ' @ : +
- & (& として受け渡しされます)

パスの正規化

CloudFront は RFC 3986 に従って URI パスを正規化し、そのパスを正しいキャッシュビヘイビアと照合します。キャッシュビヘイビアが一致すると、CloudFront は未加工の URI パスをオリジンに送信します。一致しない場合、リクエストは代わりにデフォルトのキャッシュビヘイビアと照合されます。

複数のスラッシュ (//) やピリオド (..) など、一部の文字は正規化されてパスから削除されます。これにより、CloudFront が使用する URL が意図したキャッシュビヘイビアに合わせて変更される場合があります。

Example 例

キャッシュビヘイビアの /a/b* パスと /a* パスを指定します。

- /a/b?c=1 パスを送信するビューワーは、/a/b* キャッシュビヘイビアと一致します。
- /a/b/..?c=1 パスを送信するビューワーは、/a* キャッシュビヘイビアと一致します。

パスの正規化を回避するには、リクエストパスまたはキャッシュビヘイビアのパスパターンを更新できます。

オリジンまたはオリジングループ

この設定は、既存のディストリビューションのキャッシュ動作を作成または更新する場合のみ適用されます。

既存のオリジンまたはオリジングループの値を入力します。これは、リクエスト (https://example.com/logo.jpg など) がキャッシュ動作 (*.jpg) またはデフォルトキャッシュ動作 (*) のパスパターンに一致するときに、CloudFront にリクエストをルーティングさせる宛先となるオリジンまたはオリジングループを識別します。

ビューワープロトコルポリシー

CloudFront エッジロケーションのコンテンツへのアクセスに使用するビューワーのプロトコルポリシーを選択します。

- [HTTP and HTTPS (HTTP と HTTPS)]: ビューワーは両方のプロトコルを使用できます。
- [Redirect HTTP to HTTPS (HTTP を HTTPS にリダイレクト)]: ビューワーは両方のプロトコルを使用できますが、HTTP リクエストは自動的に HTTPS リクエストにリダイレクトされます。
- [HTTPS Only (HTTPS のみ)]: ビューワーは HTTPS を使用している場合にのみコンテンツにアクセスできます。

詳細については、「ビューワーと CloudFront の間の通信に HTTPS を要求する」を参照してください。

許可される HTTP メソッド

CloudFront が処理してオリジンに転送する HTTP メソッドを指定します。

- [GET, HEAD]: CloudFront を使用して、オリジンからのオブジェクトの取得またはオブジェクトヘッダーの取得のみを行うことができます。
- [GET, HEAD, OPTIONS]: CloudFront を使用して、オリジンからのオブジェクトの取得、オブジェクトヘッダーの取得、またはオリジンサーバーがサポートするオプションのリスト取得のみを行うことができます。
- [GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE]: CloudFront を使用して、オブジェクトの取得、追加、更新、削除、およびオブジェクトヘッダーの取得を行うことができます。また、ウェブフォームからのデータの送信など、その他の POST 操作も実行できます。

> **Note**
>
> CloudFront は、GET リクエスト、HEAD リクエスト、および (オプションで) OPTIONS リクエストに対する応答をキャッシュします。OPTIONS リクエストへのレスポンスは、GET および HEAD リクエストへのレスポンスとは別にキャッシュされます (OPTIONS メソッドは、OPTIONS リクエストの [cache key] (キャッシュキー) 内にあります)。CloudFront はその他のメソッドを使用するリクエストへのレスポンスをキャッシュしません。

> **Important**
>
> [GET, HEAD, OPTIONS] または [GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE] を選択した場合は、望ましくない操作がユーザーによって実行されないように、Amazon S3 バケットまたはカスタムオリジンへのアクセスを制限する必要が生じることがあります。以下の例は、アクセスを制限する方法を示しています。
>
> - ディストリビューションのオリジンとして Amazon S3 を使用している場合: Amazon S3 コンテンツへのアクセスを制限するための CloudFront オリジンアクセスコントロールを作成し、そのオリジンアクセスコントロールにアクセス許可を付与します。例えば、PUT を使用したいというだけの理由で、上記のメソッドを受け入れて転送するように CloudFront を設定する場合は、Amazon S3 バケットのポリシーを DELETE リクエストを適切に処理するように引き続き設定する必要があります。詳しくは、「Amazon Simple Storage Service オリジンへのアクセスを制限する」を参照してください。
> - カスタムオリジンを使用している場合: すべてのメソッドを処理するようにオリジンサーバーを設定します。たとえば、POST を使用したいというだけの理由で、上記のメソッドを受け入れて転送するように CloudFront を構成するという場合は、オリジンサーバーを DELETE リクエストを適切に処理するように引き続き構成する必要があります。

フィールドレベル暗号化の設定

特定のデータフィールドにフィールドレベル暗号化を適用する場合は、ドロップダウンリストからフィールドレベル暗号化の設定を選択します。

詳細については、「フィールドレベル暗号化を使用した機密データの保護」を参照してください。

キャッシュされる HTTP メソッド

ビューワーから OPTIONS リクエストが送信されたときに、オリジンからの応答を CloudFront でキャッシュするかどうかを指定します。CloudFront は、GET リクエストと HEAD リクエストへの応答を常にキャッシュします。

選択されたリクエストヘッダーに基づいたキャッシュ

指定したヘッダーの値に基づいてオブジェクトを CloudFront でキャッシュするかどうかを指定します。

- [None (improves caching) (なし (キャッシュを改善))] – CloudFront はヘッダー値に基づいたオブジェクトのキャッシュを行いません。
- [許可リスト] – CloudFront は、指定されたヘッダー値のみに基づいてオブジェクトをキャッシュします。[許可リストヘッダー] を使って、CloudFront がキャッシュ対象とすりするヘッダーを選択します。
- [All (すべて)] – CloudFront は、このキャッシュ動作に関連付けられているオブジェクトをキャッシュしません。その代わりに、CloudFront はすべてのリクエストをオリジンに送信します (Amazon S3 オリジンには推奨されません)。

選択したオプションにかかわらず、CloudFront は特定のヘッダーをオリジンに転送し、転送するヘッダーに基づいて特定のアクションを実行します。CloudFront がヘッダーの転送を処理する方法の詳細については、「HTTP リクエストヘッダーと CloudFront の動作 (カスタムオリジンおよび Amazon S3 オリジン)」を参照してください。

リクエストヘッダーを使用した CloudFront でのキャッシュの設定方法の詳細については、「リクエストヘッダーに基づいてコンテンツをキャッシュする」を参照してください。

許可リストヘッダー

これらの設定は、[選択されたリクエストヘッダーに基づくキャッシュ] で [許可リスト] を選択した場合にのみ適用されます。

オブジェクトをキャッシュする際に CloudFront が考慮するヘッダーを指定します。使用可能なヘッダーのリストからヘッダーを選択し、[Add (追加)] を選択します。カスタムヘッダーを転送するには、フィールドにそのヘッダーの名前を入力して [Add Custom (カスタムの追加)] を選択します。

キャッシュ動作ごとに許可リストに追加できるヘッダーの現在の最大数、またはクォータ (以前は制限と呼ばれていました) の引き上げを要求するヘッダーの最大数については、「ヘッダーのクォータ」を参照してください。

オブジェクトキャッシュ

オリジンサーバーが Cache-Control ヘッダーをオブジェクトに追加して、オブジェクトを CloudFront キャッシュに保持する期間を制御している場合、Cache-Control の値を変更しないときは、[Use Origin Cache Headers (オリジンキャッシュヘッダーの使用)] を選択します。

Cache-Control ヘッダーに関係なくオブジェクトを CloudFront キャッシュに保持する最小および最大期間を指定するには、また、オブジェクトに Cache-Control ヘッダーがないときにオブジェクトを CloudFront キャッシュに保持するデフォルトの期間を指定するには、[Customize (カスタマイズ)] を選択します。次に、[Minimum TTL (最小 TTL)]、[Default TTL (デフォルト TTL)]、[Maximum TTL (最大 TTL)] の各フィールドで値を指定します。

詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

最小 TTL

オブジェクトに更新あったかどうかを照会するため、CloudFront が次のリクエストをオリジンに送信するまでの期間、オブジェクトを CloudFront キャッシュに保持しておきたい最小時間 (秒単位) を指定します。

詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

最大 TTL

オブジェクトが更新されたかどうかを CloudFront がオリジンに照会するまでに、オブジェクトを CloudFront キャッシュに保持する最大期間 (秒) を指定します。[Maximum TTL (最大 TTL)] に指定する値は、オリジンが Cache-Control max-age、Cache-Control s-maxage、Expires などの HTTP ヘッダーをオブジェクトに追加するときにのみ適用されます。詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

[Maximum TTL (最大 TTL)] の値を指定するには、[Object Caching (オブジェクトキャッシュ)] 設定で [Customize (カスタマイズ)] オプションを選択する必要があります。

[Maximum TTL (最大 TTL)] のデフォルト値は 31,536,000 (秒)、つまり 1 年です。[Minimum TTL (最小 TTL)] または [Default TTL (デフォルト TTL)] の値を 31,536,000 (秒) より大きい値に変更する場

合は、[Maximum TTL (最大 TTL)] のデフォルト値を [Default TTL (デフォルト TTL)] の値に変更します。

デフォルト TTL

オブジェクトが更新されたかどうかを調べるために CloudFront がオリジンに別のリクエストを送るまでオブジェクトを CloudFront キャッシュに保持するデフォルト期間 (秒) を指定します。[Default TTL] (デフォルト TTL) に指定する値が適用されるのは、オリジンが `Cache-Control max-age`、`Cache-Control s-maxage`、`Expires` などの HTTP ヘッダーをオブジェクトに追加しない場合のみです。詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

[Default TTL (デフォルト TTL)] の値を指定するには、[Object Caching (オブジェクトキャッシュ)] 設定で [Customize (カスタマイズ)] オプションを選択する必要があります。

[Default TTL (デフォルト TTL)] のデフォルト値は 86,400 (秒)、つまり 1 日です。[Minimum TTL (最小 TTL)] の値を 86,400 (秒) より大きい値に変更する場合は、[Default TTL (デフォルト TTL)] のデフォルト値を [Minimum TTL (最小 TTL)] の値に変更します。

cookie の転送

> **Note**
>
> Amazon S3 オリジンの場合、このオプションは、ウェブサイトエンドポイントとして設定されているバケットにのみ適用されます。

CloudFront からオリジンサーバーに Cookie を転送するかどうかと、転送する場合にどれを転送するかを指定します。選択された Cookie (Cookie の許可リスト) のみを転送するように選択した場合、Cookie 名を [許可リスト Cookie] フィールドに入力します。[All (すべて)] を選択した場合、アプリケーションで使用されている Cookie の数に関係なく、CloudFront はすべての Cookie を転送します。

Amazon S3 は、Cookie を処理しません。オリジンに Cookie を転送すると、キャッシュ能力が低下します。リクエストを Amazon S3 オリジンに転送するキャッシュ動作の場合は、[Forward Cookies (Cookie の転送)] で [None (なし)] を選択します。

オリジンへの Cookie の転送の詳細については、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。

許可リスト Cookie

> **Note**
>
> Amazon S3 オリジンの場合、このオプションは、ウェブサイトエンドポイントとして設定されているバケットにのみ適用されます。

[Cookie を転送する] リストで [許可リスト] を選択した場合は、このキャッシュ動作に応じて CloudFront がオリジンサーバーに転送する Cookie の名前を [許可リスト Cookie] フィールドに入力します。各 Cookie 名を新しい行に入力します。

以下のワイルドカード文字を使用して Cookie 名を指定することができます。

- * は、Cookie 名に含まれる 0 個以上の文字に一致します。
- ? は、Cookie 名に含まれる 1 文字に一致します。

たとえば、オブジェクトに対するビューワーリクエストに、次の名前の Cookie が含まれているとします。

`userid_`*`member-number`*

`member-number` は、各ユーザーに割り当てられた一意の値です。各メンバーについて、個別バージョンのオブジェクトを CloudFront でキャッシュするものとします。そのためにすべての Cookie をオリジンに転送することもできますが、ビューワーリクエストには、CloudFront でキャッシュすることが望ましくない Cookie も含まれています。これに代わる方法として、Cookie 名に以下の値を指定できます。その場合、CloudFront は `userid_` から始まるすべての Cookie をオリジンに転送します。

```
userid_*
```

キャッシュ動作ごとに許可リストに追加できる Cookie 名の現在の最大数、またはクォータの引き上げ (以前は制限と呼ばれていました) を要求する Cookie 名の最大数については、「Cookie のクォータ (従来のキャッシュ設定)」を参照してください。

クエリ文字列の転送とキャッシュ

CloudFront は、クエリ文字列パラメータの値に基づいて、コンテンツのさまざまなバージョンをキャッシュできます。次のいずれかのオプションを選択します。

None (Improves Caching)

オリジンがクエリ文字列パラメータの値に関係なくオブジェクトの同じバージョンを返す場合、このオプションを選択します。これにより、CloudFront がキャッシュからリクエストを処理できる可能性が高くなり、パフォーマンスが向上し、オリジンの負荷が低下します。

すべて転送、許可リストに基づいてキャッシュ

オリジンサーバーが 1 つ以上のクエリ文字列パラメータに基づいてオブジェクトの異なるバージョンを返す場合、このオプションを選択します。次に、キャッシュ条件として CloudFront が使用するパラメータを [クエリ文字列の許可リスト] フィールドに指定します。

Forward all, cache based on all

オリジンサーバーがすべてのクエリ文字列パラメータについてオブジェクトの異なるバージョンを返す場合、このオプションを選択します。

パフォーマンスを向上する方法を含む、クエリ文字列パラメータに基づくキャッシュについて詳しくは、「クエリ文字列パラメータに基づいてコンテンツをキャッシュする」を参照してください。

クエリ文字列の許可リスト

この設定は、クエリ文字列の転送とキャッシュ の [すべて転送、許可リストに基づいてキャッシュ] を選択した場合にのみ適用されます。CloudFront によってキャッシュの基準として使用されるクエリ文字列パラメータを指定できます。

スムーズストリーミング

Microsoft Smooth Streaming 形式のメディアファイルを配信するが、IIS サーバーがない場合は、[Yes (はい)] を選択します。

Microsoft Smooth Streaming 形式のメディアファイルを配信するためのオリジンとして使用する Microsoft IIS サーバーがある場合、または Smooth Streaming メディアファイルを配信しない場合は、[No (いいえ)] を選択します。

> **Note**
>
> [Yes (はい)] を指定した場合でも、他のコンテンツが [Path Pattern (パスパターン)] の値と一致すれば、このキャッシュ動作を使用してそのコンテンツを配信できます。

詳細については、「Microsoft Smooth Streaming のビデオオンデマンドを設定する」を参照してください。

ビューワーのアクセス制限 (署名付き URL または署名付き cookie の使用)

このキャッシュ動作の PathPattern に一致するオブジェクトのリクエストでパブリック URL を使用する場合、[No (いいえ)] を選択します。

このキャッシュ動作の PathPattern に一致するオブジェクトのリクエストで署名付き URL を使用する場合、[Yes (はい)] を選択します。次に、署名付き URL の作成に使用する AWS アカウントを指定します。このアカウントは信頼された署名者として知られています。

信頼された署名者の詳細については、「署名付き URL と署名付き Cookie を作成できる署名者を指定する」を参照してください。

信頼された署名者

この設定は、[ビューワーのアクセスを制限 (署名付き URL または署名付き Cookie の使用)] で [はい] を選択した場合にのみ適用されます。

このキャッシュ動作の信頼された署名者として使用する AWS アカウントを選択します。

- Self: 信頼された署名者として AWS Management Console へのサインインに現在使用しているアカウントを使用します。現在、IAM ユーザーとしてサインインしている場合は、関連付けられた AWS アカウントを、信頼された署名者として追加します。
- Specify Accounts: 信頼された署名者のアカウント番号を [AWS Account Numbers] フィールドに入力します。

署名付き URL を作成するには、AWS アカウント番号に少なくとも 1 つのアクティブな CloudFront キーペアが必要です。

> **Important**
>
> コンテンツの配信で既に使用されているディストリビューションを更新する場合は、オブジェクトの署名付き URL の生成を開始する準備ができたときにのみ、信頼された署名者を追加します。信頼された署名者がディストリビューションに追加されると、ユーザーは、このキャッシュ動作の PathPattern に一致するオブジェクトへのアクセスに、署名付き URL を使用する必要があります。

AWS アカウント番号

この設定は、[信頼された署名者] で [アカウントを指定] を選択した場合にのみ適用されます。

現在のアカウントに加えて、または現在のアカウントの代わりに、AWS アカウントを使用して署名付き URL を作成する場合、このフィールドの行ごとに 1 つの AWS アカウント番号を入力します。次の点に注意してください。

- 指定するアカウントに少なくとも 1 つのアクティブな CloudFront キーペアが必要です。詳細については、「署名者のキーペアを作成する」を参照してください。
- IAM ユーザーの CloudFront キーペアを作成できないため、信頼された署名者として IAM ユーザーを使用することはできません。
- アカウントの AWS アカウント番号を取得する方法については、「Amazon Web Services 全般のリファレンス」の「AWS アカウント ID」を参照してください。
- 現在のアカウントのアカウント番号を入力した場合、CloudFront は自動的に [Self] (セルフ) のチェックボックスをオンにして、[AWS Account Numbers] (AWS アカウント番号) リストからそのアカウント番号を削除します。

オブジェクトを自動的に圧縮する

ビューワーが圧縮コンテンツをサポートしている場合に、特定のタイプのファイルを CloudFront で自動的に圧縮するには、[Yes (はい)] を選択します。CloudFront がコンテンツを圧縮すると、ファイルが小さくなるため、ダウンロードが速くなります。また、ユーザーに対するウェブページのレンダリングが高速化されます。詳細については、「圧縮ファイルを供給する」を参照してください。

CloudFront イベント

この設定は、Lambda 関数の関連付けに適用されます。

次の CloudFront イベントが 1 つ以上発生したときは、Lambda 関数を実行することを選択できます。

- CloudFront がビューワーからリクエストを受信したとき (ビューワーリクエスト)
- CloudFront がリクエストをオリジンに転送する前 (オリジンリクエスト)
- CloudFront がオリジンからレスポンスを受信したとき (オリジンレスポンス)
- CloudFront がビューワーにレスポンスを返す前 (ビューワーレスポンス)

詳細については、「どの CloudFront イベントを使用して Lambda@Edge 関数をトリガーするかを決定する」を参照してください。

Lambda 関数の ARN

この設定は、Lambda 関数の関連付けに適用されます。

トリガーを追加する Lambda 関数の Amazon リソースネーム (ARN) を指定します。関数の ARN を取得する方法については、「CloudFront コンソールを使ってトリガーを追加する」の手順のステップ 1 を参照してください。

本文を含める

この設定は、Lambda 関数の関連付けに適用されます。

詳細については、「ボディを含める」を参照してください。

ディストリビューションの設定

以下の値はディストリビューション全体に適用されます。

トピック

- 価格クラス
- AWS WAF ウェブ ACL
- 代替ドメイン名 (CNAME)
- SSL 証明書
- 独自 SSL クライアントのサポート
- セキュリティポリシー (SSL/TLS の最小バージョン)
- サポートされる HTTP バージョン
- デフォルトのルートオブジェクト
- ログ記録
- ログ用のバケット
- ログのプレフィックス
- cookie のログ作成
- IPv6 を有効にする
- コメント
- ディストリビューションの状態

価格クラス

CloudFront サービスに支払う上限価格に対応する価格クラスを選択します。デフォルトでは、CloudFront は、すべての CloudFront リージョンのエッジロケーションからオブジェクトを供給します。

料金クラスと、選択した料金クラスがディストリビューションの CloudFront パフォーマンスに与える影響の詳細については、「CloudFront の料金」を参照してください。

AWS WAF ウェブ ACL

CloudFront ディストリビューションは、ウェブアプリケーションと API を保護してリクエストがサーバーに到達する前にブロックできるようにするウェブアプリケーションファイアウォールである AWS WAF で保護できます。CloudFront ディストリビューションを作成または編集するときに、ディストリビューションで AWS WAF を有効にすることができます。

オプションで、後で AWS WAF コンソール (https://console.aws.amazon.com/wafv2/) から、アプリケーションに固有の他の脅威に対する追加のセキュリティ保護を設定できます。

AWS WAF の詳細については、AWS WAF 開発者ガイドを参照してください。

代替ドメイン名 (CNAME)

オプション。ディストリビューションを作成するときに CloudFront が割り当てるドメイン名ではなく、オブジェクトの URL に使用する 1 つ以上のドメイン名を指定します。ドメイン名を所有しているか、あるいはこのドメイン名を使用する許可があることが必要です。この許可は、SSL/TLS 証明書を追加することで検証します。

たとえば、次のオブジェクトの URL があります。

```
/images/image.jpg
```

この URL を次のように表示します。

```
https://www.example.com/images/image.jpg
```

次のようには指定しません。

```
https://d111111abcdef8.cloudfront.net/images/image.jpg
```

この場合、`www.example.com` の CNAME を追加します。

> **Important**
>
> ディストリビューションに `www.example.com` の CNAME を追加する場合、さらに以下を実行する必要があります。
>
> - DNS サービスを使用して CNAME レコードを作成 (または更新) して、`www.example.com` のクエリを `d111111abcdef8.cloudfront.net` にルーティングします。
> - ディストリビューションに追加するドメイン名 (CNAME) が対象の信頼される認証機関 (CA) から証明書を CloudFront に追加して、このドメイン名を使用する許可を検証します。
>
> ドメインの DNS サービスプロバイダーがある CNAME レコードを作成する許可が必要です。通常、これはドメインを所有している、またはドメイン所有者向けにアプリケーションを開発していることを示します。

ディストリビューションに対して作成できる代替ドメイン名の現在の最大数、またはクォータの引き上げを要求する代替ドメイン名の最大数については、「ディストリビューションの一般的なクォータ」を参照してください。

代替ドメイン名の詳細については、「代替ドメイン名 (CNAME) を追加することによって、カスタム URL を使用する」を参照してください。CloudFront URL の詳細については、「CloudFront でファイルの URL 形式をカスタマイズする」を参照してください。

SSL 証明書

ディストリビューションで使用する代替ドメイン名を指定した場合は、[Custom SSL Certificate (カスタム SSL 証明書)] を選択してこの代替ドメイン名を使用する許可を検証し、これを対象とする証明書を選択します。ビューワーが HTTPS を使用してオブジェクトにアクセスするようにする場合は、それをサポートしている設定を選択します。

> **Note**
>
> カスタム SSL 証明書を指定する前に、有効な代替ドメイン名を指定する必要があります。詳細については、代替ドメイン名を使用するための要件および代替ドメイン名と HTTPS を使用するを参照してください。

- デフォルトの CloudFront 証明書 (*.cloudfront.net) – オブジェクトの URL で `https://d111111abcdef8.cloudfront.net/image1.jpg` のような CloudFront ドメイン名を使用する場合は、このオプションを選択します。
- 独自 SSL 証明書 – オブジェクトの URL で独自のドメイン名を代替ドメイン名として使用する場合 (`https://example.com/image1.jpg` など)、このオプションを選択します。次に、代替ドメイン名を対象とする証明書を使用するために選択します。証明書のリストには、次のいずれかを含めることができます。
  - AWS Certificate Manager から提供される証明書
  - サードパーティー認証機関から購入して ACM にアップロードした証明書
  - サードパーティー認証機関から購入して IAM 証明書ストアにアップロードした証明書

  この設定を選択した場合、オブジェクト URL でのみ代替ドメイン名を使用することをお勧めします (https://example.com/logo.jpg)。CloudFront ディストリビューションドメイン名 (https://d111111abcdef8.cloudfront.net/logo.jpg) を使用し、クライアントが SNI をサポートしない古いビューワーを使用している場合、ビューワーが応答する方法は [Clients Supported (サポートされるクライアント)] で選択した値に応じて異なります。
  - [All Clients (すべてのクライアント)]: CloudFront ドメイン名が SSL/TLS 証明書のドメイン名と一致しないため、ビューワーには警告が表示されます。
  - [Only Clients that Support Server Name Indication (SNI) (Server Name Indication (SNI) をサポートするクライアントのみ)]: CloudFront はオブジェクトを返さないでビューワーとの接続を中断します。

独自 SSL クライアントのサポート

[SSL 証明書] で [カスタム SSL 証明書 (example.com)] を選択した場合にのみ適用されます。ディストリビューションに 1 つ以上の代替ドメイン名と独自 SSL 証明書を指定した場合は、CloudFront が HTTPS リクエストを処理する方法を選択します。

- [Clients that Support Server Name Indication (SNI) - (Recommended) (Server Name Indication (SNI) をサポートするクライアント - (推奨))] – この設定では、ほとんどすべての最新のウェブブラウザとクライアントは、SNI をサポートしているため、ディストリビューションに接続できます。ただし、一部のビューワーは SNI をサポートしていない古いウェブブラウザやクライアントを使用している可能性があります。つまり、一部のビューワーはディストリビューションに接続できません。

CloudFront API を使用してこの設定を適用するには、`SSLSupportMethod` フィールドで `sni-only` を指定します。AWS CloudFormation では、このフィールドの名前は `SslSupportMethod` です (大文字と小文字の変更に注意してください)。

- [Legacy Clients Support (レガシークライアントサポート)] – この設定では、SNI をサポートしていない古いウェブブラウザとクライアントはディストリビューションに接続できます。ただし、この設定では、追加の月額料金が発生します。正確な価格については、[Amazon CloudFront Pricing (Amazon CloudFront の料金)] ページに移動し、[Dedicated IP custom SSL (専用 IP 独自 SSL)] のページを検索します。

  CloudFront API を使用してこの設定を適用するには、`SSLSupportMethod` フィールドで `vip` を指定します。AWS CloudFormation では、このフィールドの名前は `SslSupportMethod` です (大文字と小文字の変更に注意してください)。

詳細については、「CloudFront で HTTPS リクエストを処理する方法を選択する」を参照してください。

セキュリティポリシー (SSL/TLS の最小バージョン)

CloudFront がビューワー (クライアント) との HTTPS 接続に使用するセキュリティポリシーを指定します。セキュリティポリシーは 2 通りの設定を決定します。

- CloudFront でビューワーとの通信に使用する最小の SSL/TLS プロトコル。
- ビューワーに返すコンテンツを暗号化するために CloudFront で使用できる暗号。

セキュリティポリシー (各ポリシーに含まれるプロトコルや暗号など) の詳細については、「ビューワーと CloudFront との間でサポートされているプロトコルと暗号」を参照してください。

利用可能なセキュリティポリシーは、[SSL Certificate (SSL 証明書)] および [Custom SSL Client Support (独自 SSL クライアントサポート)] に指定する値によって異なります (CloudFront API では `CloudFrontDefaultCertificate` および `SSLSupportMethod` が該当します)。

- [SSL Certificate (SSL 証明書)] が [Default CloudFront Certificate (デフォルトの CloudFront 証明書) (*.cloudfront.net)] である場合 (API では `CloudFrontDefaultCertificate` が `true` である場合)、CloudFront はセキュリティポリシーを自動的に TLSv1 に設定します。
- [SSL Certificate] (SSL 証明書) が [Custom SSL Certificate (example.com)] (カスタム SSL 証明書) で、かつ [Custom SSL Client Support] (カスタム SSL クライアントサポート) が [Clients that Support Server Name Indication (SNI) - (Recommended)] (Server Name Indication (SNI) をサポー

トするクライアント) - (推奨)、(つまり API で `CloudFrontDefaultCertificate` が `false` かつ `SSLSupportMethod` が `sni-only` である場合) 次のセキュリティポリシーから選択できます。

- TLSv1.2_2021
- TLSv1.2_2019
- TLSv1.2_2018
- TLSv1.1_2016
- TLSv1_2016
- TLSv1

- [SSL Certificate] (SSL 証明書) が [Custom SSL Certificate (example.com)] (カスタム SSL 証明書) で、かつ [Custom SSL Client Support] (カスタム SSL クライアントサポート) が [Legacy Clients Support] (レガシークライアントサポート)、(つまり API で `CloudFrontDefaultCertificate` が `false` かつ `SSLSupportMethod` が `vip` である場合) 次のセキュリティポリシーから選択できます。

  - TLSv1
  - SSLv3

  この設定では、TLSv1.2_2021、TLSv1.2_2019、TLSv1.2_2018、TLSv1.1_2016、および TLSv1_2016 の各セキュリティポリシーを CloudFront コンソールまたは API で使用することはできません。これらのセキュリティポリシーのいずれかを使用する場合は、以下のオプションを指定できます。

  - ディストリビューションが専用 IP アドレスを使用したレガシークライアントサポートを必要としているかどうかを評価します。ビューワーが [Server Name Indication (SNI)] (Server Name Indication (SNI)) をサポートしている場合は、ディストリビューションの [Custom SSL Client Support] (カスタム SSL クライアントサポート) の設定を [Clients that Support Server Name Indication (SNI)] (Server Name Indication (SNI)) をサポートするクライアント に更新 (API で `SSLSupportMethod` を `sni-only` に設定する) することをお勧めします。これにより、利用可能な TLS セキュリティポリシーのいずれでも使用できます。また、CloudFront の料金も削減できます。
  - 専用 IP アドレスを使用するレガシークライアントのサポートを維持する必要がある場合は、AWS サポートセンターでケースを作成することによって、他の TLS セキュリティポリシー (TLSv1.2_2021、TLSv1.2_2019、TLSv1.2_2018、TLSv1.1_2016、または TLSv1_2016) のいずれかをリクエストできます。

> **Note**
>
> AWS サポートに連絡してこの変更をリクエストする前に、以下の点を考慮してください。
>
> - これらのセキュリティポリシー (TLSv1.2_2021、TLSv1.2_2019、TLSv1.2_2018、TLSv1.1_2016、または TLSv1_2016) のいずれかをレガシークライアントサポートディストリビューションに追加すると、セキュリティポリシーが AWS アカウントのすべてのレガシークライアントサポートディストリビューションに対する SNI 以外のすべてのビューワーリクエストに適用されます。ただし、ビューワーが、レガシークライアントサポートを使用するディストリビューションに SNI リクエストを送信した場合は、そのディストリビューションのセキュリティポリシーが適用されます。AWS アカウントのすべてのレガシークライアントサポートディストリビューションに送信されたすべてのビューワーリクエストに望ましいセキュリティポリシーが適用されることを確実にするには、望ましいセキュリティポリシーをディストリビューションごとに個別に追加します。
> - 定義上、新しいセキュリティポリシーは、古いセキュリティポリシーと同じ暗号やプロトコルをサポートしません。たとえば、ディストリビューションのセキュリティポリシーを TLSv1 から TLSv1.1_2016 にアップグレードすることを選択した場合、そのディストリビューションは DES-CBC3-SHA 暗号をサポートしなくなります。各セキュリティポリシーがサポートする暗号とプロトコルの詳細については、「ビューワーと CloudFront との間でサポートされているプロトコルと暗号」を参照してください。

サポートされる HTTP バージョン

ビューワーが CloudFront と通信するときにディストリビューションでサポートする HTTP バージョンを選択します。

ビューワーと CloudFront が HTTP/2 を使用するには、ビューワーが TLSv1.2 以降と Server Name Indication (SNI) をサポートしている必要があります。CloudFront は HTTP/2 経由で gRPC のネイティブサポートを提供していません。

ビューワーと CloudFront が HTTP/3 を使用するには、ビューワーが TLSv1.2 以降と Server Name Indication (SNI) をサポートしている必要があります。CloudFront は、ビューワーが接続を失わずに

ネットワークを切り替えることができるように、HTTP/3 接続の移行をサポートしています。接続の移行の詳細については、RFC 9000 で「Connection Migration」を参照してください。

> **Note**
>
> サポートされる TLSv1.3 暗号の詳細については、「ビューワーと CloudFront との間でサポートされているプロトコルと暗号」を参照してください。

デフォルトのルートオブジェクト

オプション。ビューワーがディストリビューション内のオブジェクト (`index.html`) ではなくディストリビューションのルート URL (`https://www.example.com/`) をリクエストするときに、CloudFront がリクエストする、オリジンからのオブジェクト (`https://www.example.com/product-description.html` など)。デフォルトのルートオブジェクトを指定すると、ディストリビューションのコンテンツが公開されなくなります。

名前の最大長は 255 文字です。名前には以下の文字を含めることができます。

- A ~ Z、a ~ z
- 0-9
- _ - . * $ / ~ " '
- & (& として受け渡しされます)

デフォルトのルートオブジェクトを指定する場合、オブジェクト名のみを指定します (例: `index.html`)。オブジェクト名の前に / を追加しないでください。

詳細については、「デフォルトのルートオブジェクトを指定する」を参照してください。

ログ記録

CloudFront がオブジェクトに対する各リクエストの情報をログに記録し、ログファイルを Amazon S3 バケットに保存するかどうか。ログ作成はいつでも有効または無効にできます。ログ作成を有効にしても追加料金はかかりませんが、Amazon S3 バケットにおけるファイルの保存とファイルへのアクセスについては通常の Amazon S3 料金が発生します。ログの削除はいつでも行うことができます。CloudFront アクセスログの詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

ログ用のバケット

[Logging] (ログ) で [On] (オン) を選択した場合に、CloudFront がアクセスログを保存する Amazon S3 バケット (例: *myLogs*-DOC-EXAMPLE-BUCKET.s3.amazonaws.com)。

> **Important**
>
> 以下のいずれのリージョンでも Amazon S3 バケットを選択しないでください。CloudFront は、これらのリージョンのバケットに標準ログを配信しません。
>
> - アフリカ (ケープタウン)
> - アジアパシフィック (香港)
> - アジアパシフィック (ハイデラバード)
> - アジアパシフィック (ジャカルタ)
> - アジアパシフィック (メルボルン)
> - カナダ西部 (カルガリー)
> - 欧州 (ミラノ)
> - 欧州 (スペイン)
> - 欧州 (チューリッヒ)
> - イスラエル (テルアビブ)
> - 中東 (バーレーン)
> - 中東 (アラブ首長国連邦)

ログ作成を有効にした場合、CloudFront はオブジェクトに対する各エンドユーザーリクエストの情報を記録し、ファイルを、指定された Amazon S3 バケットに保存します。ログ作成はいつでも有効または無効にできます。CloudFront アクセスログの詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

> **Note**
>
> Amazon S3 バケット ACL を取得して更新するための権限が必要です。また、バケットの S3 ACL から FULL_CONTROL が付与される必要があります。これによって、CloudFront から awslogsdelivery アカウントに、バケットにログファイルを保存するための権限が付与さ

れます。詳細については、「標準ログ記録の設定およびログファイルへのアクセスに必要なアクセス許可」を参照してください。

ログのプレフィックス

オプション。[Logging (ログ)] で [On (オン)] を選択した場合、このディストリビューションのアクセスログファイル名の先頭に CloudFront が追加する文字列 (ある場合) を指定します (例: `exampleprefix/`)。末尾のスラッシュ (/) はオプションですが、ログファイルの参照を容易にするためにこれを使用することをお勧めします。CloudFront アクセスログの詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

cookie のログ作成

CloudFront で Cookie をアクセスログに含めるようにするには、[On (オン)] を選択します。Cookie をログに含めるように選択した場合、CloudFront はすべての Cookie をログに記録します。このディストリビューションのキャッシュ動作がどのように構成されているか (オリジンにすべての Cookie を転送するか、Cookie を転送しないか、指定された一連の Coolie を転送するか) は関係ありません。

Amazon S3 は Cookie を処理しません。したがって、ディストリビューションに Amazon EC2 または他のカスタムオリジンも含まれていない限り、[Cookie Logging (Cookie ログ記録)] の値に [Off (オフ)] を選択することをお勧めします。

Cookie の詳細については、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。

IPv6 を有効にする

IPv6 は、IP プロトコルの新しいバージョンです。これは IPv4 に今後取って代わるもので、より大きなアドレス空間を使用します。CloudFront は常に IPv4 リクエストに応答します。IPv4 の IP アドレス (192.0.2.44 など) からのリクエストや IPv6 のアドレス (2001:0db8:85a3::8a2e:0370:7334 など) からのリクエストに CloudFront が応答できるようにするには、[Enable IPv6] (IPv6 の有効化) を選択します。

一般的に、IPv6 ネットワークのユーザーがいてコンテンツにアクセスする場合は、IPv6 を有効にする必要があります。ただし、コンテンツへのアクセスを制限するために署名付き URL または署名付き Cookie を使用していて、コンテンツへのアクセスが可能な IP アドレスを制限する IP アドレスを

含む IpAddress パラメータを使用している場合、IPv6 は有効にしません。一部のコンテンツへのアクセスを IP アドレスで制限し、他のコンテンツへのアクセスを制限しない場合 (またはアクセスを制限するが IP アドレスでは行わない場合)、2 つのディストリビューションを作成します。カスタムポリシーを使用して署名付き URL を作成する方法については、カスタムポリシーを使用して署名付き URL を作成するを参照してください。カスタムポリシーを使用して署名付き Cookie を作成する方法については、カスタムポリシーを使用して署名付き Cookie を設定するを参照してください。

Route 53 エイリアスリソースレコードセットを使用して CloudFront ディストリビューションにトラフィックをルーティングしている場合、次の両方に該当するときには、2 つ目のエイリアスリソースレコードセットを作成する必要があります。

- ディストリビューションで IPv6 を有効にする
- オブジェクトの URL で代替ドメイン名を使用している

詳細については、『Amazon Route 53 開発者ガイド』の「ドメイン名を使用したトラフィックの Amazon CloudFront ディストリビューションへのルーティング」を参照してください。

Route 53 または別の DNS サービスで CNAME リソースレコードセットを作成した場合、変更を行う必要はありません。ビューワーリクエストの IP アドレスフォーマットに関係なく、CNAME レコードはトラフィックをディストリビューションにルーティングします。

IPv6 と CloudFront アクセスログを有効にすると、c-ip 列には IPv4 および IPv6 フォーマットの値が含められます。詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

> **Note**
>
> お客様に対する優れた可用性を維持するために、データによって IPv4 がより優れたユーザーエクスペリエンスを提供することが判明している場合、CloudFront は IPv4 を使用してビューワーリクエストに応じます。CloudFront が IPv6 によって対応するリクエストの割合を知るには、ディストリビューションで CloudFront ロギングを有効にし、リクエストを行ったビューワーの IP アドレスを含む c-ip 列を解析します。この割合 (%) は時間とともに大きくなりますが、IPv6 は世界中のすべてのビューワーネットワークでサポートされているわけではないため、過半数のトラフィックになることはないでしょう。ビューワーネットワークによっては IPv6 が十分にサポートされていますが、IPv6 をまったくサポートしないものもあります (ビューワーネットワークは、ホームインターネットやワイヤレスキャリアに似ています)。

IPv6 のサポートについては、「CloudFront に関するよくある質問」を参照してください。アクセスログを有効にする方法の詳細については、ログ記録、ログ用のバケット、および ログのプレフィックス のフィールドを参照してください。

コメント

(オプション)。ディストリビューションを作成するときに、最大で 128 文字のコメントを含めることができます。コメントの更新はいつでも行うことができます。

ディストリビューションの状態

ディストリビューションがデプロイされた後にディストリビューションを有効または無効のどちらにするかを示します。

- Enabled (有効): ディストリビューションが完全にデプロイされると、ディストリビューションのドメイン名を使用するリンクをデプロイでき、ユーザーはコンテンツを取得できます。ディストリビューションを有効にすると、CloudFront はそのディストリビューションに関連付けられたドメイン名を使用するコンテンツへのエンドユーザーリクエストを受け付けて処理します。

  CloudFront ディストリビューションの作成、変更、削除を行った場合、その変更が CloudFront データベースに伝達されるまで時間がかかります。変更直後に出したディストリビューションに関する情報のリクエストには、変更が反映されていない可能性があります。通常、伝達は数分以内で完了しますが、システムの高負荷またはネットワークパーティションによっては、それより時間がかかる可能性があります。
- Disabled (無効): ディストリビューションがデプロイされていて、使用準備ができていても、ユーザーはディストリビューションを使用できません。ディストリビューションを無効にするときはいつでも、そのディストリビューションに関連付けられたドメイン名を使用するコンテンツへのエンドユーザーリクエストを CloudFront は受け付けません。(ディストリビューションの構成を更新することで) ディストリビューションを無効から有効に切り替えるまで、誰もディストリビューションを使用できません。

ディストリビューションの無効と有効は何度でも切り替えることができます。ディストリビューションの構成を更新するプロセスに従います。詳細については、「ディストリビューションを更新する」を参照してください。

カスタムエラーページとエラーキャッシュ

Amazon S3 またはカスタムオリジンが HTTP 4xx または 5xx ステータスコードを CloudFront に返す場合、CloudFront にオブジェクトをビューワーに返させることができます (例: HTML ファイル)。オリジンまたはカスタムエラーページからのエラーレスポンスを CloudFront エッジキャッシュにキャッシングする時間を指定することもできます。詳細については、「HTTP ステータスコード別のカスタムエラーページを作成する」を参照してください。

> **Note**
>
> 以下の値は [Create Distribution] ウィザードに含まれていないため、ディストリビューションを更新するときにのみ、カスタムエラーページを構成することができます。

トピック

HTTP エラーコード

CloudFront がカスタムエラーページを返す HTTP ステータスコード。CloudFront がキャッシュする HTTP ステータスコードの全部または一部に対応するカスタムエラーページを返すように、または全く返さないように、CloudFront を構成できます。

Response page path (レスポンスページのパス)

[Error Code (エラーコード)] で指定した HTTP ステータスコード (403 など) がオリジンから返されたときに、CloudFront がビューワーに返すカスタムエラーページのパス (例: `/4xx-errors/403-forbidden.html`)。オブジェクトとカスタムエラーページを別の場所に保存する場合は、次の状況に該当するときに適用されるキャッシュ動作をディストリビューションに組み込む必要があります。

- [Path Pattern (パスパターン)] の値が、カスタムエラーメッセージのパスと一致している。たとえば、4xx エラーのカスタムエラーページを `/4xx-errors` というディレクトリの Amazon S3 バケットに保存したとします。このとき、パスパターンによってカスタムエラーページのリクエスト

のルーティング先である場所に対するキャッシュ動作を、ディストリビューションに組み込む必要があります (例: /4xx-errors/*)。

- [Origin (オリジン)] の値は、カスタムエラーページが含まれているオリジンの [Origin ID (オリジン ID)] の値を指定しています。

HTTP レスポンスコード

CloudFront からカスタムエラーページとともにビューワーに返す HTTP ステータスコード。

Error caching minimum TTL (seconds) (エラーキャッシュ最小 TTL (秒))

CloudFront がオリジンサーバーからのエラーレスポンスをキャッシュする最小時間。

地理的制限

特定の国のユーザーにコンテンツへのアクセスを禁止する必要がある場合は、CloudFront ディストリビューションで [許可リスト] または [ブロックリスト] を設定できます。地理的制限の設定には追加料金が発生しません。詳細については、「コンテンツの地理的配分を制限する」を参照してください。

ディストリビューションのテスト

ディストリビューションの作成が終わると、CloudFront にオリジンサーバーの場所が記憶され、ドメイン名がディストリビューションに関連付けられます。ディストリビューションをテストするには、次の手順を実行します。

1. ディストリビューションがデプロイされるまで待ちます。
   - コンソールでディストリビューションの詳細を表示します。ディストリビューションのデプロイが完了すると、[最終変更日] フィールドが [デプロイ中] から 日付と時刻に変わります。
2. 次の手順を使用して、CloudFront ドメイン名でオブジェクトへのリンクを作成します。
3. リンクをテストします。CloudFront は、ウェブページまたはアプリケーションにオブジェクトを提供します。

オブジェクトへのリンクを作成する

CloudFront ウェブディストリビューションでオブジェクトのテストリンクを作成するには、次の手順に従います。

ウェブディストリビューション内のオブジェクトへのリンクを作成するには

1. 以下の HTML コードを新しいファイルにコピーし、*domain-name* をディストリビューションのドメイン名で置き換え、*object-name* をオブジェクトの名前で置き換えます。

   ```
   <html>
   <head>My CloudFront Test</head>
   <body>
   <p>My text content goes here.</p>
   <p><img src="https://domain-name/object-name" alt="my test image"
   </body>
   </html>
   ```

   たとえば、ドメイン名が d111111abcdef8.cloudfront.net で、オブジェクトが image.jpg の場合、リンクの URL は次のようになります。

   https://d111111abcdef8.cloudfront.net/image.jpg.

   オブジェクトがオリジンサーバー内のフォルダにある場合は、そのフォルダも URL に含める必要があります。たとえば、image.jpg がオリジンサーバーのイメージフォルダにある場合、URL は次のようになります。

   https://d111111abcdef8.cloudfront.net/images/image.jpg
2. HTML コードを、html ファイル名拡張子の付いたファイルに保存します。
3. ブラウザでウェブページを開いて、オブジェクトが見られるかどうかを確認します。

ブラウザに、イメージファイルが埋め込まれたページが表示されます。これは CloudFront がオブジェクトを供給するのに適切と判断したエッジロケーションから供給されたものです。

ディストリビューションを更新する

CloudFront コンソールで、AWS アカウントに関連付けられた CloudFront ディストリビューションを表示したり、ディストリビューションの設定を表示したり、ほとんどの設定を更新したりできます。設定に変更を加えても、ディストリビューションが AWS エッジロケーションに伝達されるまで有効にならないことに注意してください。

CloudFront ディストリビューションを更新するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ディストリビューションの ID を選択します。リストには、CloudFront コンソールへのサインインに使用した AWS アカウントに関連付けられているすべてのディストリビューションが含まれています。
3. ディストリビューションの設定を編集するには、[Distribution Settings (ディストリビューションの設定)] タブを選択します。
4. 全般設定を更新するには、[Edit (編集)] を選択します。それ以外の場合は、更新する設定のタブ ([Origins (オリジン)] または [Behaviors (動作)]) を選択します。
5. 更新してから変更内容を保存するには、[Yes, Edit (はい、編集します)] を選択します。フィールドの詳細については、以下の各トピックを参照してください。
   - 全般設定: ディストリビューションの設定
   - オリジンの設定: オリジンの設定
   - キャッシュ動作の設定: キャッシュ動作の設定
6. ディストリビューションのオリジンを削除する場合は、次の操作を行います。
   a. [Behaviors (動作)] を選択し、そのオリジンに関連付けられているデフォルトのキャッシュ動作が別のオリジンに移動済みであることを確認します。
   b. [Origins (オリジン)] で、オリジンを選択します。
   c. [Delete] (削除) を選択します。

CloudFront API を使用してディストリビューションを更新することもできます。

- ディストリビューションを更新するには、『Amazon CloudFront API リファレンス』の「UpdateDistribution」を参照してください。

> **Important**
>
> ディストリビューションを更新する際は、ディストリビューションの作成時には必要でないいくつかの追加フィールドが必須であることに注意してください。CloudFront API を使用してディストリビューションを更新するときにすべての必須フィールドを確実に含めるには、

「Amazon CloudFront API リファレンス」の「UpdateDistribution」で説明している手順に従います。

ディストリビューション構成に対する変更を保存すると、CloudFront はすべてのエッジロケーションへの変更の伝達を開始します。連続した設定の変更は、それぞれの順序で伝播されます。エッジロケーションで構成が更新されるまで、CloudFront は以前の構成に基づいて、そのエッジロケーションからコンテンツを引き続き供給します。エッジロケーションで構成が更新されると、CloudFront は新しい構成に基づいて、そのエッジロケーションからコンテンツの供給を直ちに開始します。

変更は、すべてのエッジロケーションに同時に伝達されるわけではありません。CloudFront が変更を伝播している間は、特定のエッジロケーションで以前の設定と新しい設定のどちらに基づいてコンテンツを提供しているかを判断することはできません。

変更がいつ伝達されるかを確認するには、コンソールでディストリビューションの詳細を表示します。デプロイが完了すると、[最終変更日] フィールドが [デプロイ中] から日付と時刻に変わります。

ディストリビューションのタグ付け

タグは、AWS リソースを特定し、整理するのに使用できる単語または語句です。各リソースには複数のタグを追加でき、各タグにはユーザーが定義したキーと値が含まれています。たとえば、キーが "domain" で値が "example.com" というタグを付けることができます。追加したタグに基づいて、リソースを検索したりフィルタ処理したりできます。

以下の例のように、CloudFront でタグを使用できます。

- CloudFront ディストリビューションにタグベースのアクセス許可を適用します。詳細については、「CloudFront での ABAC」を参照してください。
- さまざまなカテゴリの請求情報を追跡します。CloudFront ディストリビューションまたは AWS リソース (Amazon EC2 インスタンスや Amazon S3 バケットなど) にタグを適用してタグをアクティブ化すると、AWS がアクティブなタグごとに使用量とコストを集計したコスト配分レポートを CSV (カンマ区切り値) ファイルとして生成します。

  自社のカテゴリ たとえばコストセンター、アプリケーション名、所有者を表すタグを適用すると、複数のサービスにわたってコストを分類することができます。タグを使ったコスト配分の詳細については、AWS Billing ユーザーガイドのコスト配分タグの使用を参照してください。

メモ

- ディストリビューションをタグ付けできますが、オリジンアクセスアイデンティティや無効化をタグ付けすることはできません。
- タグエディタおよび リソースグループは、現在、CloudFront でサポートされていません。
- ディストリビューションに追加できるタグの数の現在の最大制限については、「一般的なクォータ」を参照してください。

目次

タグの制限

タグには以下のベーシックな制限があります。

- ディストリビューションあたりのタグの最大数については、「一般的なクォータ」を参照してください。
- キーの最大長 – 128 文字 (Unicode)
- 値の最大長 – 256 文字 (Unicode)
- キーと値の有効な値 – a ~ z、A ~ Z、0 ~ 9、スペース、特殊文字 (_ . : / = + - @)
- タグのキーと値は大文字と小文字が区別されます。
- aws: をキーのプレフィックスとして使用しないでください。このプレフィックスは AWS 専用として予約されています。

ディストリビューションに対するタグの追加、編集、削除

CloudFront コンソールを使用して、ディストリビューションのタグを管理できます。

ディストリビューションにタグを追加、編集、または削除するには

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

2. 更新するディストリビューションの ID を選択します。
3. [タグ] タブを選択します。
4. [タグを管理] を選択します。
5. [Manage tags] (タグの管理) ページで、次の操作を実行できます。
   - タグを追加するには、キーと、タグの値 (オプション) を入力します。タグをさらに追加するには、[新しいタグを追加] を選択します。
   - タグを編集するには、タグのキーまたはその値、あるいはその両方を変更します。タグの値を削除することはできますが、キーが必要です。
   - タグを削除するには、[削除] を選択します。
6. [Save changes] (変更の保存) をクリックします。

プログラムによるタグ付け

CloudFront API、AWS Command Line Interface (AWS CLI)、AWS SDK、および AWS Tools for Windows PowerShell を使用して、リソースにタグを適用することもできます。詳細については、次のトピックを参照してください。

- CloudFront API 操作:
  - ListTagsForResource
  - TagResource
  - UntagResource
- AWS CLI –AWS CLI コマンドリファレンスの「cloudfront」を参照
- AWS SDK – 「AWS ドキュメント」ページの該当する SDK ドキュメントを参照
- Tools for Windows PowerShell — AWS Tools for PowerShell Cmdlet リファレンスにある「Amazon CloudFront」を参照してください

ディストリビューションを削除する

以下の手順では、CloudFront コンソールを使用してディストリビューションを削除します。CloudFront API を使用した削除については、「Amazon CloudFront API リファレンス」の「DeleteDistribution」を参照してください。

S3 バケットに OAC がアタッチされているディストリビューションを削除する必要がある場合は、重要な詳細について「S3 バケットに OAC がアタッチされたディストリビューションを削除する」を参照してください。

> **Note**
>
> ディストリビューションを削除する前に、これを無効にする必要があることに注意してください。そのためには、ディストリビューションを更新するアクセス許可が必要です。
> 代替ドメイン名が関連付けられているディストリビューションを無効にすると、CloudFront は、別のディストリビューションにこのドメイン (*.example.com) と一致するワイルドカード (*) 付きの代替ドメイン名がある場合でも、このドメイン名 (たとえば、www.example.com) へのトラフィックの受信を停止します。

CloudFront ディストリビューションを削除するには

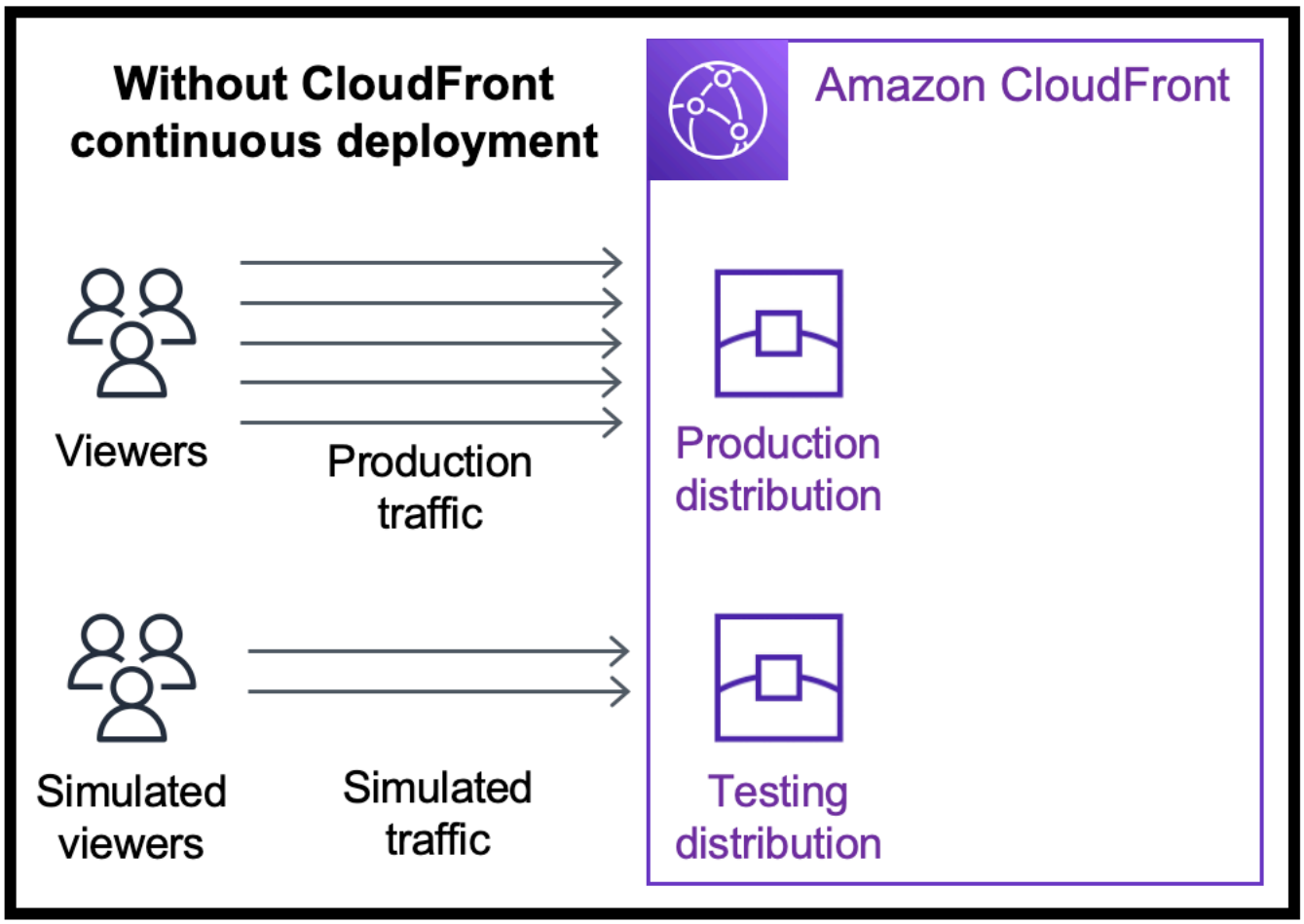
1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. CloudFront コンソールの右のペインで、削除するディストリビューションを見つけます。
   - [ステータス] 列に [無効] と表示されている場合は、ステップ 6 に進みます。
   - [ステータス] に [有効] と表示されているのに、[最終変更日] 列にディストリビューションが [デプロイ中] と表示される場合は、デプロイが完了するのを待ってからステップ 3 に進みます。
3. CloudFront コンソールの右のペインで、削除するディストリビューションのチェックボックスをオンにします。
4. [Disable (無効)] を選択してディストリビューションを無効にし、[Yes, Disable (はい、無効にします)] を選択して確定します。次に、[Close] (閉じる) を選択します。
   - [ステータス] 列の値がすぐに [無効] に変わります。
5. [最終変更日] 列に新しいタイムスタンプが表示されるまで待ちます。
   - CloudFront がすべてのエッジロケーションに変更を伝達するまでに数分かかる場合があります。
6. 削除するディストリビューションのチェックボックスをオンにします。
7. [Delete] (削除) を選択し、削除します。

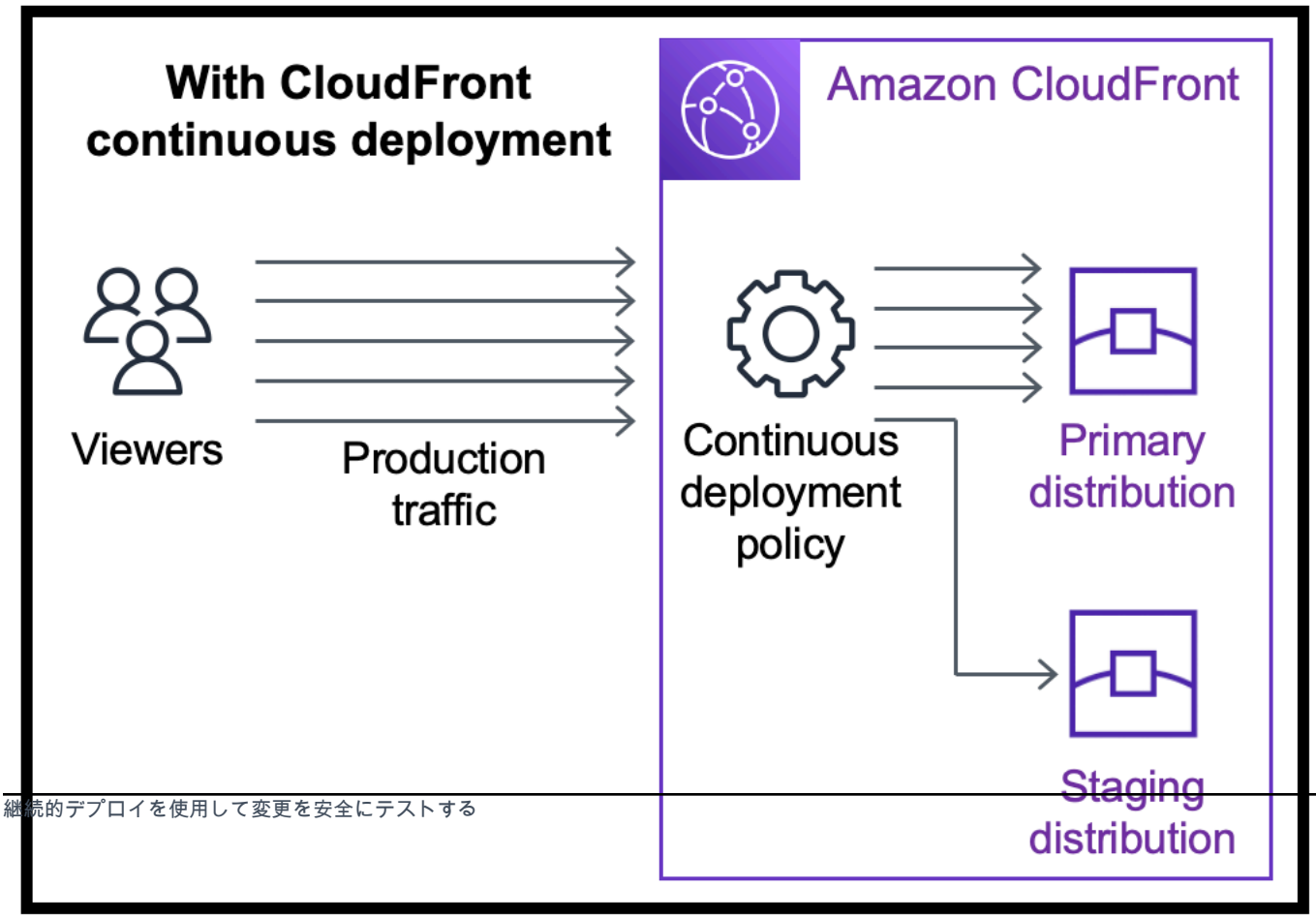
- [削除] オプションが使用できない場合、CloudFront がエッジロケーションに変更を伝達中であることを意味します。[最終変更日] 列に新しいタイムスタンプが表示されるまで待ってから、ステップ 6 ~ 7 を繰り返します。

CloudFront の継続的デプロイを使用して CDN 設定の変更を安全にテストする

Amazon CloudFront の継続的デプロイを使用すると、最初に本番トラフィックのサブセットでテストすることで、CDN 設定に変更を安全にデプロイできます。ステージングディストリビューションと継続的デプロイポリシーを使用して、実際 (本番環境) のビューワーからのトラフィックを新しい CDN 設定に送信し、正常に動作することを検証できます。新しい設定のパフォーマンスをリアルタイムでモニタリングし、準備ができたら、新しい設定を昇格させてすべてのトラフィックをプライマリディストリビューションで処理できます。

次の図は、CloudFront の継続的デプロイを使用するメリットを示しています。これを使用しない場合は、シミュレートしたトラフィックで CDN 設定の変更をテストする必要があります。継続的デプロイでは、本番トラフィックのサブセットで変更をテストし、準備ができたら、変更をプライマリディストリビューションに昇格させることができます。


Without CloudFront continuous deployment
Amazon CloudFront
Viewers
Production traffic
Production distribution
Simulated viewers
Simulated traffic
Testing distribution


With CloudFront continuous deployment
Amazon CloudFront
Viewers
Production traffic
Continuous deployment policy
Primary distribution
Staging distribution

継続的デプロイの使用の詳細については、以下のトピックを参照してください。

トピック

- CloudFront の継続的デプロイのワークフロー
- ステージングディストリビューションと継続的デプロイポリシーを使用する
- ステージングディストリビューションをモニタリングする
- 継続的デプロイの仕組みについて説明します。
- 継続的デプロイに関するクォータとその他の考慮事項

CloudFront の継続的デプロイのワークフロー

次の高レベルのワークフローでは、CloudFront の継続的デプロイを使用して、設定の変更を安全にテストしてデプロイする方法を説明しています。

1. プライマリディストリビューションとして使用するディストリビューションを選択します。プライマリーディストリビューションは、本番トラフィックを現在処理しているディストリビューションです。
2. プライマリディストリビューションから、ステージングディストリビューションを作成します。ステージングディストリビューションは、プライマリディストリビューションのコピーとして開始します。
3. 継続的デプロイポリシー内にトラフィック設定を作成し、それをプライマリディストリビューションにアタッチします。これにより、CloudFront がトラフィックをステージングディストリビューションにルーティングする方法が決まります。ステージングディストリビューションにリクエストをルーティングする方法の詳細については、「the section called “ステージングディストリビューションにリクエストをルーティングする”」を参照してください。
4. ステージングディストリビューションの設定を更新します。更新できる設定の詳細については、「the section called “プライマリーディストリビューションとステージングディストリビューションを更新する”」を参照してください。
5. ステージングディストリビューションをモニタリングして、設定の変更が正常に動作するかどうかを確認します。ステージングディストリビューションのモニタリングの詳細については、「the section called “ステージングディストリビューションをモニタリングする”」を参照してください。

   ステージングディストリビューションをモニタリングしながら、以下のことを実行できます。

- ステージングディストリビューションの設定を再度更新し、設定の変更のテストを続行します。
- 継続的デプロイポリシー (トラフィック設定) を更新して、ステージングディストリビューションに送信するトラフィックを増減します。

6. ステージングディストリビューションのパフォーマンスに満足したら、ステージングディストリビューションの設定をプライマリディストリビューションに昇格させます。これにより、ステージングディストリビューションの設定がプライマリディストリビューションにコピーされます。これに伴って、継続的デプロイポリシーが無効になるため、CloudFront はすべてのトラフィックをプライマリディストリビューションにルーティングします。

オートメーションを構築することで、ステージングディストリビューションのパフォーマンスをモニタリングして (ステップ 5)、特定の基準を満たしたときに自動的に設定を昇格させる (ステップ 6) ことができます。

設定を昇格させると、設定の変更を次回テストするときに、同じステージングディストリビューションを再利用できます。

CloudFront コンソール、AWS CLI、または CloudFront API でのステージングディストリビューションと継続的デプロイポリシーの使用方法の詳細については、次のセクションを参照してください。

ステージングディストリビューションと継続的デプロイポリシーを使用する

ステージングディストリビューションと継続的デプロイポリシーは、CloudFront コンソール、AWS Command Line Interface (AWS CLI)、または CloudFront API で作成、更新、変更できます。

ステージングディストリビューションと継続的デプロイポリシーを作成する

次の手順では、ステージングディストリビューションと継続的デプロイポリシーを作成する方法を示します。

Console

AWS Management Console を使用して、ステージングディストリビューションと継続的デプロイポリシーを作成できます。

ステージングディストリビューションと継続的デプロイポリシーを作成するには (コンソール)

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[ディストリビューション] を選択します。
3. プライマリディストリビューションとして使用するディストリビューションを選択します。プライマリーディストリビューションは、本番トラフィックを現在処理しているディストリビューションであり、これを元にしてステージングディストリビューションを作成します。
4. [Continuous deployment] (継続的デプロイ) セクションで、[Create staging distribution] (ステージングディストリビューションを作成) を選択します。[Create staging distribution] (ステージングディストリビューションを作成) ウィザードが開きます。
5. [Create staging distribution] (ステージングディストリビューションを作成) ウィザードで、以下の操作を行います。

   a. (オプション) ステージングディストリビューションの説明を入力します。

   b. [Next] (次へ) を選択します。

   c. ステージングディストリビューションの設定を変更します。更新できる設定の詳細については、「the section called “プライマリーディストリビューションとステージングディストリビューションを更新する”」を参照してください。

      ステージングディストリビューションの設定の変更が完了したら、[Next] (次へ) を選択します。

   d. コンソールを使用して [Traffic configuration] (トラフィック設定) を指定します。これにより、CloudFront がトラフィックをステージングディストリビューションにルーティングする方法が決まります (CloudFront はトラフィック設定を継続的デプロイポリシー内に保存します)。

      [Traffic configuration] (トラフィック設定) のオプションの詳細については、「the section called “ステージングディストリビューションにリクエストをルーティングする”」を参照してください。

      [Traffic configuration] (トラフィック設定) が完了したら、[Next] (次へ) を選択します。

   e. トラフィック設定を含む、ステージングディストリビューションの設定を確認したら、[Create staging distribution] (ステージングディストリビューションを作成) を選択します。

CloudFront コンソールで [Create staging distribution] (ステージングディストリビューションを作成) ウィザードを完了すると、CloudFront は以下を実行します。

- ステップ 5c で指定した設定を使用して、ステージングディストリビューションを作成します。
- ステップ 5d で指定したトラフィック設定を使用して、継続的デプロイポリシーを作成します。
- ステージングディストリビューションの作成元のプライマリディストリビューションに継続的デプロイポリシーをアタッチします。

継続的デプロイポリシーをアタッチした、プライマリディストリビューションの設定をエッジロケーションにデプロイすると、CloudFront はトラフィック設定に基づいてトラフィックの指定された部分をステージングディストリビューションに送信し始めます。

CLI

ステージングディストリビューションと継続的デプロイポリシーを AWS CLI で作成するには、次の手順に従います。

ステージングディストリビューションを作成するには (CLI)

1. aws cloudfront get-distribution コマンドと grep コマンドを一緒に使用して、プライマリディストリビューションとして使用するディストリビューションの ETag 値を取得します。プライマリーディストリビューションは、本番トラフィックを現在処理しているディストリビューションであり、これを元にしてステージングディストリビューションを作成します。

   次のコマンドでは、例を示しています。次の例では、*primary_distribution_ID* をプライマリディストリビューションの ID に置き換えます。

   ```
   aws cloudfront get-distribution --id primary_distribution_ID | grep 'ETag'
   ```

   ETag 値をコピーします (次のステップで必要になります)。

2. aws cloudfront copy-distribution コマンドを使用してステージングディストリビューションを作成します。次のコマンド例では、読みやすくするためにエスケープ文字 (\) と改行を使用していますが、これらはコマンドから省略してください。次のコマンドの例で以下の操作を行います。

- *primary_distribution_ID* をプライマリディストリビューションの ID に置き換えます。
- *primary_distribution_ETag* をプライマリディストリビューションの ETag 値 (前のステップで取得したもの) に置き換えます。
- (オプション) *CLI_example* を目的の発信者リファレンス ID に置き換えます。

```
aws cloudfront copy-distribution --primary-distribution-
id primary_distribution_ID \
                                 --if-match primary_distribution_ETag \
                                 --staging \
                                 --caller-reference 'CLI_example'
```

コマンドの出力に、ステージングディストリビューションとその設定に関する情報が表示されます。ステージングディストリビューションの CloudFront ドメイン名をコピーします (次のステップで必要になります)。

継続的デプロイポリシーを作成するには (CLI および入力ファイル)

1. 次のコマンドを使用して、continuous-deployment-policy.yaml コマンドのすべての入力パラメータを含む、create-continuous-deployment-policy という名前のファイルを作成します。次のコマンドでは、読みやすくするためにエスケープ文字 (\) と改行を使用していますが、これらはコマンドから省略してください。

```
aws cloudfront create-continuous-deployment-policy --generate-cli-skeleton yaml-
input \
                                                    > continuous-deployment-
policy.yaml
```

2. 先ほど作成した continuous-deployment-policy.yaml という名前のファイルを開きます。このファイルを編集して、必要な継続的デプロイポリシー設定を指定し、ファイルを保存します。ファイルは以下のように編集します。

   - StagingDistributionDnsNames セクションでの編集
     - Quantity の値を 1 に変更します。

- Items に、ステージングディストリビューションの CloudFront ドメイン名 (前のステップで保存したもの) を貼り付けます。
- TrafficConfig セクションでの編集
  - Type として、SingleWeight または SingleHeader を選択します。
  - 他のタイプの設定を削除します。例えば、重みベースのトラフィック設定が必要な場合は、Type を SingleWeight に設定し、SingleHeaderConfig 設定を削除します。
  - 重みベースのトラフィック設定を使用するには、Weight の値を .01 (1%) から .15 (15%) までの 10 進数に設定します。

  TrafficConfig のオプションの詳細については、「the section called “ステージングディストリビューションにリクエストをルーティングする”」および「the section called “重みベースの設定におけるセッションの維持”」を参照してください。

3. 次のコマンドで continuous-deployment-policy.yaml ファイルの入力パラメータを使用し、継続的デプロイポリシーを作成します。

```
aws cloudfront create-continuous-deployment-policy --cli-input-yaml file://
continuous-deployment-policy.yaml
```

   コマンドの出力の Id 値をコピーします。これは継続的デプロイポリシー ID で、次のステップで必要になります。

継続的デプロイポリシーをプライマリディストリビューションにアタッチするには (CLI および入力ファイル)

1. 次のコマンドを使用して、プライマリディストリビューションの設定を primary-distribution.yaml という名前のファイルに保存します。*primary_distribution_ID* をプライマリディストリビューションの ID に置き換えます。

```
aws cloudfront get-distribution-config --id primary_distribution_ID --output
 yaml > primary-distribution.yaml
```

2. 先ほど作成した primary-distribution.yaml という名前のファイルを開きます。ファイルを編集し、以下の変更を加えます。

- 継続的デプロイポリシー ID (前のステップでコピーしたもの) を `ContinuousDeploymentPolicyId` フィールドに貼り付けます。
- `ETag` フィールドの名前を `IfMatch` に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. 次のコマンドを使用して、継続的デプロイポリシーを使用するようにプライマリディストリビューションを更新します。*`primary_distribution_ID`* をプライマリディストリビューションの ID に置き換えます。

```
aws cloudfront update-distribution --id primary_distribution_ID --cli-input-yaml
 file://primary-distribution.yaml
```

継続的デプロイポリシーをアタッチした、プライマリディストリビューションの設定をエッジロケーションにデプロイすると、CloudFront はトラフィック設定に基づいてトラフィックの指定された部分をステージングディストリビューションに送信し始めます。

API

CloudFront API を使用してステージングポリシーと継続的デプロイポリシーを作成するには、以下の API オペレーションを使用します。

- CopyDistribution
- CreateContinuousDeploymentPolicy

これらの API コールで指定するフィールドの詳細については、以下を参照してください。

- the section called “ステージングディストリビューションにリクエストをルーティングする”
- the section called “重みベースの設定におけるセッションの維持”
- AWS SDK またはその他の API クライアントの API リファレンスドキュメント

ステージングディストリビューションと継続的デプロイポリシーを作成したら、プライマリディストリビューションで UpdateDistribution を使用して、継続的デプロイポリシーをプライマリディストリビューションにアタッチします。

ステージングディストリビューションを更新する

次の手順では、ステージングディストリビューションと継続的デプロイポリシーを更新する方法を示します。

Console

プライマリディストリビューションとステージングディストリビューションの両方で特定の設定を更新できます。詳細については、「プライマリーディストリビューションとステージングディストリビューションを更新する」を参照してください。

ステージングディストリビューションを更新するには (コンソール)

1. https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[ディストリビューション] を選択します。
3. プライマリディストリビューションを選択します。これは、本番トラフィックを現在処理しているディストリビューションで、ステージングディストリビューションの作成元のディストリビューションです。
4. [View staging distribution] (ステージングディストリビューションを表示) を選択します。
5. コンソールを使用して、ステージングディストリビューションの設定を変更します。更新できる設定の詳細については、「the section called “プライマリーディストリビューションとステージングディストリビューションを更新する”」を参照してください。

ステージングディストリビューションの設定をエッジロケーションにデプロイするとすぐに、ステージングディストリビューションにルーティングされた受信トラフィックに設定が適用されます。

CLI

ステージングディストリビューションを更新するには (CLI および入力ファイル)

1. 次のコマンドを使用して、ステージングディストリビューションの設定を `staging-distribution.yaml` という名前のファイルに保存します。*`staging_distribution_ID`* をステージングディストリビューションの ID に置き換えます。

```
aws cloudfront get-distribution-config --id staging_distribution_ID --output
 yaml > staging-distribution.yaml
```

2. 先ほど作成した staging-distribution.yaml という名前のファイルを開きます。ファイルを編集し、以下の変更を加えます。

   - ステージングディストリビューションの設定を変更します。更新できる設定の詳細については、「the section called “プライマリーディストリビューションとステージングディストリビューションを更新する”」を参照してください。
   - ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。

   完了したら、ファイルを保存します。

3. 次のコマンドを使用して、ステージングディストリビューションの設定を更新します。*staging_distribution_ID* をステージングディストリビューションの ID に置き換えます。

```
aws cloudfront update-distribution --id staging_distribution_ID --cli-input-yaml
 file://staging-distribution.yaml
```

ステージングディストリビューションの設定をエッジロケーションにデプロイするとすぐに、ステージングディストリビューションにルーティングされた受信トラフィックに設定が適用されます。

API

ステージングディストリビューションの設定を更新するには、ステージングディストリビューションで UpdateDistribution を使用して、ステージングディストリビューションの設定を変更します。更新できる設定の詳細については、「the section called “プライマリーディストリビューションとステージングディストリビューションを更新する”」を参照してください。

継続的デプロイポリシーを更新する

次の手順では、継続的デプロイポリシーを更新する方法を示します。

Console

継続的デプロイポリシーを更新することで、ディストリビューションのトラフィック設定を更新できます。

継続的デプロイポリシーを更新するには (コンソール)

1. https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[ディストリビューション] を選択します。
3. プライマリディストリビューションを選択します。これは、本番トラフィックを現在処理しているディストリビューションで、ステージングディストリビューションの作成元のディストリビューションです。
4. [Continuous deployment] (継続的デプロイ) セクションで、[Edit policy] (ポリシーの編集) を選択します。
5. 継続的デプロイポリシーのトラフィック設定を変更します。完了したら、[変更を保存] を選択します。

継続的デプロイポリシーを更新した、プライマリディストリビューションの設定をエッジロケーションにデプロイすると、CloudFront は更新したトラフィック設定に基づいてステージングディストリビューションにトラフィックを送信し始めます。

CLI

継続的デプロイポリシーを更新するには (CLI および入力ファイル)

1. 次のコマンドを使用して、継続的デプロイポリシーの設定を `continuous-deployment-policy.yaml` という名前のファイルに保存します。*`continuous_deployment_policy_ID`* を継続的デプロイポリシーの ID に置き換えます。次のコマンドでは、読みやすくするために エスケープ文字 (\) と改行を使用していますが、これらはコマンドから省略してください。

```
aws cloudfront get-continuous-deployment-policy-config --
id continuous_deployment_policy_ID \
                                                    --output yaml >
 continuous-deployment-policy.yaml
```

2. 先ほど作成した `continuous-deployment-policy.yaml` という名前のファイルを開きます。ファイルを編集し、以下の変更を加えます。

   - 継続的デプロイポリシーの設定を必要に応じて変更します。例えば、トラフィック設定をヘッダーベースから重みベースに変更したり、重みベースの設定でトラフィックの割合 (重み) を変更したりできます。詳細については、「the section called “ステージングディ

ストリビューションにリクエストをルーティングする”」および「the section called “重みベースの設定におけるセッションの維持”」を参照してください。

- ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. 次のコマンドを使用して、継続的デプロイポリシーを更新します。*continuous_deployment_policy_ID* を継続的デプロイポリシーの ID に置き換えます。次のコマンドでは、読みやすくするためにエスケープ文字 (\) と改行を使用していますが、これらはコマンドから省略してください。

```
aws cloudfront update-continuous-deployment-policy --
id continuous_deployment_policy_ID \
                                                --cli-input-yaml file://
continuous-deployment-policy.yaml
```

継続的デプロイポリシーを更新した、プライマリディストリビューションの設定をエッジロケーションにデプロイすると、CloudFront は更新したトラフィック設定に基づいてステージングディストリビューションにトラフィックを送信し始めます。

API

継続的デプロイポリシーを更新するには、UpdateContinuousDeploymentPolicy を使用します。

ステージングディストリビューション設定を昇格させる

次の手順では、ステージングディストリビューション設定を昇格させる方法を示します。

Console

ステージングディストリビューションを昇格させると、CloudFront はステージングディストリビューションからプライマリディストリビューションに設定をコピーします。また、CloudFront は継続的デプロイポリシーを無効にし、すべてのトラフィックをプライマリディストリビューションにルーティングします。

設定を昇格させると、設定の変更を次回テストするときに、同じステージングディストリビューションを再利用できます。

ステージングディストリビューションの設定を昇格させるには (コンソール)

1. https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[ディストリビューション] を選択します。
3. プライマリディストリビューションを選択します。これは、本番トラフィックを現在処理しているディストリビューションで、ステージングディストリビューションの作成元のディストリビューションです。
4. [Continuous deployment] (継続的デプロイ) セクションで、[Promote] (昇格) を選択します。
5. 「**`confirm`**」と入力して [Promote] (昇格) を選択します。

CLI

ステージングディストリビューションを昇格させると、CloudFront はステージングディストリビューションからプライマリディストリビューションに設定をコピーします。また、CloudFront は継続的デプロイポリシーを無効にし、すべてのトラフィックをプライマリディストリビューションにルーティングします。

設定を昇格させると、設定の変更を次回テストするときに、同じステージングディストリビューションを再利用できます。

ステージングディストリビューションの設定を昇格させるには (CLI)

- aws cloudfront update-distribution-with-staging-config コマンドを使用して、ステージングディストリビューションの設定をプライマリディストリビューションに昇格させます。次のコマンド例では、読みやすくするために エスケープ文字 (\) と改行を使用していますが、これらはコマンドから省略してください。次のコマンドの例で以下の操作を行います。
  - *`primary_distribution_ID`* をプライマリディストリビューションの ID に置き換えます。
  - *`staging_distribution_ID`* をステージングディストリビューションの ID に置き換えます。
  - *`primary_distribution_ETag`* と *`staging_distribution_ETag`* をプライマリディストリビューションとステージングディストリビューションの ETag 値に置き換えます。例に示すように、プライマリディストリビューションの値が最初になっていることを確認します。

```
aws cloudfront update-distribution-with-staging-config --
id primary_distribution_ID \
                                                    --staging-distribution-
id staging_distribution_ID \
                                                    --if-match
 'primary_distribution_ETag,staging_distribution_ETag'
```

API

ステージングディストリビューションの設定をプライマリディストリビューションに昇格させるには、UpdateDistributionWithStagingConfig を使用します。

ステージングディストリビューションをモニタリングする

ステージングディストリビューションのパフォーマンスをモニタリングするには、CloudFront がすべてのディストリビューションに提供しているのと同じメトリクス、ログ、レポートを使用できます。例:

- CloudFront コンソールでデフォルトの CloudFront ディストリビューションメトリクス (合計リクエスト数やエラー率など) を表示できます。また、追加料金を支払うことで、ステータスコード別のキャッシュヒットレートやエラー率など、追加のメトリクスを有効にすることができます。これらのメトリクスに基づいてアラームを作成することもできます。
- 標準ログとリアルタイムログを表示して、ステージングディストリビューションが受信したリクエストに関する詳細情報を取得できます。標準ログは primary-distribution-id フィールドと primary-distribution-dns-name フィールドを示します。この 2 つのフィールドは、CloudFront からステージングディストリビューションにルーティングする前のリクエストが最初に送信された先のプライマリディストリビューションを特定するのに役立ちます。
- CloudFront コンソールでレポート (キャッシュ統計レポートなど) を表示およびダウンロードできます。

継続的デプロイの仕組みについて説明します。

以下のトピックでは、CloudFront の継続的デプロイの仕組みについて説明します。

トピック

- ステージングディストリビューションにリクエストをルーティングする
- 重みベースの設定におけるセッションの維持

- プライマリーディストリビューションとステージングディストリビューションを更新する
- プライマリディストリビューションとステージングディストリビューションはキャッシュを共有しない

ステージングディストリビューションにリクエストをルーティングする

CloudFront の継続的デプロイを使用する場合、ビューワーリクエストについては何も変更する必要がありません。ビューワーは、DNS 名、IP アドレス、または CNAME を使用してステージングディストリビューションにリクエストを直接送信することはできません。代わりに、ビューワーはプライマリ (本番稼働) ディストリビューションにリクエストを送信します。CloudFront は、継続的デプロイポリシーのトラフィック設定に基づいて、これらのリクエストの一部をステージングディストリビューションにルーティングします。トラフィック設定には次の 2 つの種類があります。

重みベース

重みベースの設定では、ビューワーリクエストの指定された割合をステージングディストリビューションにルーティングします。重みベースの設定を使用する場合、セッションの維持を有効にすることもできます。これにより、CloudFront は同じビューワーからのリクエストを 1 つのセッションの一部として処理できるようになります。詳細については、「the section called “重みベースの設定におけるセッションの維持”」を参照してください。

ヘッダーベース

ビューワーリクエストに特定の HTTP ヘッダーが含まれている (ヘッダーと値が指定されている) 場合、ヘッダーベースの設定は、リクエストをステージングディストリビューションにルーティングします。ヘッダーと値が指定されていないリクエストは、プライマリディストリビューションにルーティングされます。この設定は、ローカルでテストする場合や、ビューワーリクエストが制御可能である場合に便利です。

> **Note**
>
> ステージングディストリビューションにルーティングするヘッダーには、プレフィックス `aws-cf-cd-` が含まれている必要があります。

重みベースの設定におけるセッションの維持

重みベースの設定を使用してトラフィックをステージングディストリビューションにルーティングする場合、セッションの維持を有効にすることもできます。これにより、CloudFront は同じビュー

ワーからのリクエストを 1 つのセッションの一部として処理できるようになります。セッションの維持を有効にすると、CloudFront は Cookie を設定して、1 つのセッション内における同じビューワーからのすべてのリクエストを 1 つのディストリビューション (プライマリまたはステージング) で処理します。

セッションの維持を有効にするときに、アイドル期間を指定することもできます。ビューワーのアイドル (リクエストを送信しない) 状態がこの期間を超えると、セッションは期限切れになり、CloudFront はこのビューワーからの以降のリクエストを新しいセッションとして扱います。アイドル期間は 300 (5 分) から 3,600 (1 時間) までの秒数で指定します。

以下の場合、CloudFront はすべてのセッション (アクティブなセッションも含む) をリセットし、すべてのリクエストを新しいセッションと見なします。

- 継続的デプロイポリシーを無効または有効にする。
- セッションの維持の設定を無効または有効にする。

プライマリーディストリビューションとステージングディストリビューションを更新する

プライマリディストリビューションに継続的デプロイポリシーがアタッチされている場合、プライマリーディストリビューションとステージングディストリビューションの両方で、以下の設定の変更が可能です。

- すべてのキャッシュ動作設定 (デフォルトのキャッシュ動作を含む)
- すべてのオリジン設定 (オリジンとオリジングループ)
- カスタムエラーレスポンス (エラーページ)
- 地理的制限
- デフォルトのルートオブジェクト
- ログ記録の設定
- 説明 (コメント)

ディストリビューションの設定で参照される外部リソース (キャッシュポリシー、レスポンスヘッダーポリシー、CloudFront 関数、Lambda@Edge 関数など) を更新することもできます。

プライマリディストリビューションとステージングディストリビューションはキャッシュを共有しない

プライマリディストリビューションとステージングディストリビューションはキャッシュを共有しません。CloudFront が最初のリクエストをステージングディストリビューションに送信するまで、キャッシュは空です。リクエストがステージングディストリビューションに到着すると、レスポンスのキャッシュが開始されます (そのように設定している場合)。

継続的デプロイに関するクォータとその他の考慮事項

CloudFront の継続的デプロイには、以下のクォータとその他の考慮事項が適用されます。

クォータ

- AWS アカウント あたりのステージングディストリビューションの最大数: 20
- AWS アカウント あたりの継続的デプロイポリシーの最大数: 20
- 重みベースの設定でステージングディストリビューションに送信できるトラフィックの最大割合: 15%
- セッションの維持のアイドル期間の最小値と最大値: 300 ~ 3,600 秒

詳細については、「クォータ」を参照してください。

> **Note**
>
> 継続的デプロイを使用しており、プライマリディストリビューションが S3 バケットアクセス用の OAC で設定されている場合は、S3 バケットポリシーを更新してステージングディストリビューションへのアクセスを許可します。S3 バケットポリシーの例については、「the section called “S3 バケットへのアクセス許可をオリジンアクセスコントロールに付与する”」を参照してください。

AWS WAF ウェブ ACL

ディストリビューションで継続的ディストリビューションを有効にする場合、AWS WAF には次の考慮事項が適用されます。

- AWS WAF ウェブアクセスコントロールリスト (ACL) をディストリビューションに初めて関連付けることはできません。

- ディストリビューションから AWS WAF ウェブ ACL の関連付けを解除することはできません。

上記のタスクを実行する前に、本番稼働用ディストリビューションの継続的デプロイポリシーを削除する必要があります。これにより、ステージングディストリビューションも削除されます。詳細については、「AWS WAF 保護を使用する」を参照してください。

CloudFront がすべてのリクエストをプライマリディストリビューションに送信する場合

リソース使用率が高い期間など、状況に応じて、CloudFront は継続的デプロイポリシーで指定されている内容に関係なく、すべてのリクエストをプライマリディストリビューションに送信する場合があります。

CloudFront は、継続的デプロイポリシーで指定されている内容に関係なく、トラフィックのピーク時には、すべてのリクエストをプライマリディストリビューションに送信します。ピークトラフィックとは、CloudFront サービスのトラフィックであり、ディストリビューションのトラフィックではありません。

HTTP/3

HTTP/3 をサポートするディストリビューションでは継続的デプロイを使用できません。

CloudFront ディストリビューションでさまざまなオリジンを使用する

ディストリビューションを作成するときに、CloudFront によってファイルのリクエストが送信される場所であるオリジンを指定します。CloudFront では、さまざまなオリジンを使用できます。例えば、Amazon S3 バケット、MediaStore コンテナ、MediaPackage チャネル、Application Load Balancer、AWS Lambda 関数 URL を使用できます。

トピック

- Amazon S3 バケットを使用する
- MediaStore コンテナまたは MediaPackage チャネルを使用する
- Application Load Balancer を使用する
- Lambda 関数 URL を使用する
- Amazon EC2 (または別のカスタムオリジン) を使用する
- CloudFront オリジングループを使用する

Amazon S3 バケットを使用する

以下のトピックでは、CloudFront ディストリビューションのオリジンとして Amazon S3 バケットを使用するさまざまな方法について説明します。

トピック

標準的な Amazon S3 バケットを使用する

ディストリビューションのオリジンとして Amazon S3 を使用する場合、CloudFront が配信するオブジェクトを、Amazon S3 バケットに配置します。オブジェクトを Amazon S3 に保存するには、Amazon S3 でサポートされる任意の方法を使用できます。例えば、Amazon S3 コンソールや API、サードパーティーのツールを使用できます。他の標準 Amazon S3 バケットと同様に、バケット内に階層を作成してオブジェクトを保存できます。

既存の Amazon S3 バケットを CloudFront オリジンサーバーとして使用してもバケットに変更は一切ありません。Amazon S3 オブジェクトの保存やアクセスで通常使用しているとおりに通常の Amazon S3 価格でバケットを使用できます。バケットへのオブジェクトの保存には、通常の Amazon S3 料金が発生します。CloudFront の使用料の詳細については、「Amazon CloudFront 料金表」を参照してください。CloudFront での既存の S3 バケットの使用の詳細については、「the section called “既存の Amazon S3 バケットに CloudFront を追加する”」を参照してください。

> **Important**
>
> バケットが CloudFront で正常に機能するには、その名前が DNS 命名要件に沿ったものでなければなりません。詳細については、Amazon Simple Storage Service ユーザーガイドの「バケットの命名規則」を参照してください。

CloudFront のオリジンとして Amazon S3 バケットを指定する場合は、次の形式を使用することをお勧めします。

bucket-name.s3.*region*.amazonaws.com

この形式でバケット名を指定した場合、以下の CloudFront 機能を使用することができます。

- SSL/TLS を使用して Amazon S3 バケットと通信するように CloudFront を設定します。詳しくは、「the section called “CloudFront で HTTPS を使用する”」を参照してください。
- オリジンアクセスコントロールを使用して、Amazon S3 URL ではなく CloudFront URL を使ってコンテンツにアクセスするようビューワーに要求します。詳しくは、「the section called “Amazon Simple Storage Service オリジンへのアクセスを制限する”」を参照してください。
- POST リクエストと PUT リクエストを CloudFront に送信して、バケットのコンテンツを更新します。詳細については、トピック「the section called “CloudFront がリクエストを処理して Amazon S3 オリジンに転送する方法”」の「the section called “HTTP メソッド”」を参照してください。

以下の形式を使用してバケットを指定しないでください。

- Amazon S3 パススタイル: s3.amazonaws.com/*bucket-name*
- Amazon S3 の CNAME

Amazon S3 Object Lambda を使用する

Object Lambda アクセスポイントを作成すると、Amazon S3 は Object Lambda アクセスポイントの固有のエイリアスを自動的に生成します。CloudFront 配信のオリジンとして、Amazon S3 バケット名の代わりにこのエイリアス使用することができます。

Object Lambda アクセスポイントエイリアスを CloudFront のオリジンとして使用する場合は、次の形式を使用することをお勧めします。

alias.s3.*region*.amazonaws.com

alias の検索の詳細については、「Amazon S3 ユーザーガイド」の「S3 バケット Object Lambda アクセスポイントにおけるバケット形式のエイリアスの使用」を参照してください。

> **Important**
>
> Object Lambda アクセスポイントを CloudFront のオリジンとして使用する場合は、オリジンアクセスコントロールを使用する必要があります。

ユースケースの例については、「Amazon S3 Object Lambda を Amazon CloudFront で使用して、エンドユーザー向けにコンテンツをカスタマイズする」を参照してください。

CloudFront は、Object Lambda アクセスポイントのオリジンを標準の Amazon S3 バケットオリジンと同じように扱います。

Amazon S3 Object Lambda をディストリビューションのオリジンとして使用する場合は、次の 4 つのアクセス許可を設定する必要があります。

Object Lambda Access Point

Object Lambda アクセスポイントのアクセス許可を追加するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 左のナビゲーションペインで、[Object Lambda アクセスポイント] を選択します。
3. 使用する Object Lambda アクセスポイントを選択します。
4. [アクセス許可] タブを選択します。
5. [Object Lambda アクセスポイントポリシー] セクションで [編集] を選択します。
6. 以下のポリシーを [ポリシー] フィールドに貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3-object-lambda:Get*",
            "Resource": "arn:aws:s3-object-lambda:region:AWS-account-
ID:accesspoint/Object-Lambda-Access-Point-name",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:cloudfront::AWS-account-
ID:distribution/CloudFront-distribution-ID"
                }
            }
        }
    ]
```

```
}
```

7. [Save changes] (変更の保存) をクリックします。

Amazon S3 Access Point

Amazon S3 アクセスポイントのアクセス許可を追加するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、[アクセスポイント] を選択します。
3. 使用する Amazon S3 アクセスポイントを選択します。
4. [アクセス許可] タブを選択します。
5. [アクセスポイントポリシー] セクションで [編集] を選択します。
6. 以下のポリシーを [ポリシー] フィールドに貼り付けます。

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "s3objlambda",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:region:AWS-account-ID:accesspoint/Access-Point-
name",
                "arn:aws:s3:region:AWS-account-ID:accesspoint/Access-Point-name/
object/*"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "s3-object-lambda.amazonaws.com"
                }
            }
        }
    ]
```

```
}
```

7. [Save] を選択します。

Amazon S3 bucket

Amazon S3 バケットにアクセス許可を追加するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、バケットを選択します。
3. 使用する Amazon S3 バケットを選択します。
4. [アクセス許可] タブを選択します。
5. [バケットポリシー] セクションで、[編集] を選択します。
6. 以下のポリシーを [ポリシー] フィールドに貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "*",
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:DataAccessPointAccount": "AWS-account-ID"
                }
            }
        }
    ]
}
```

7. [Save changes] (変更の保存) をクリックします。

AWS Lambda function

Lambda 関数にアクセス許可を追加するには

1. AWS Management Console にサインインして AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. ナビゲーションペインで、[関数] を選択します。
3. 使用する AWS Lambda 関数を選択します。
4. [設定] タブを開き、次に [アクセス許可] をクリックします。
5. [リソースベースのポリシーステートメント] セクションで [アクセス許可の追加] を選択します。
6. [AWS アカウント] を選択します。
7. [ステートメント ID] の名前を入力します。
8. [プリンシパル] の `cloudfront.amazonaws.com` を入力します。
9. [アクション] ドロップダウンメニューから `lambda:InvokeFunction` を選択します。
10. [Save] を選択します。

Amazon S3 Access Point を使用する

S3 アクセスポイントを使用する場合、Amazon S3 は固有のエイリアスを自動的に生成します。CloudFront 配信のオリジンとして、Amazon S3 バケット名の代わりにこのエイリアス使用することができます。

Amazon S3 アクセスポイントエイリアスを CloudFront のオリジンとして使用する場合は、次の形式を使用することをお勧めします。

alias.s3.*region*.amazonaws.com

alias の検索の詳細については、「Amazon S3 ユーザーガイド」の「S3 バケットアクセスポイントでのバケット形式のエイリアスの使用」を参照してください。

> **Important**
>
> Amazon S3 アクセスポイントを CloudFront のオリジンとして使用する場合は、オリジンアクセスコントロールを使用する必要があります。

CloudFront は、Amazon S3 アクセスポイントオリジンを標準の Amazon S3 バケットオリジンと同じように扱います。

Amazon S3 Object Lambda をディストリビューションのオリジンとして使用する場合は、次の 2 つのアクセス許可を設定する必要があります。

Amazon S3 Access Point

Amazon S3 アクセスポイントのアクセス許可を追加するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、[アクセスポイント] を選択します。
3. 使用する Amazon S3 アクセスポイントを選択します。
4. [アクセス許可] タブを選択します。
5. [アクセスポイントポリシー] セクションで [編集] を選択します。
6. 以下のポリシーを [ポリシー] フィールドに貼り付けます。

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "s3objlambda",
            "Effect": "Allow",
            "Principal": {"Service": "cloudfront.amazonaws.com"},
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:region:AWS-account-ID:accesspoint/Access-Point-
name",
                "arn:aws:s3:region:AWS-account-ID:accesspoint/Access-Point-name/
object/*"
            ],
            "Condition": {
                "StringEquals": {"aws:SourceArn": "arn:aws:cloudfront::AWS-
account-ID:distribution/CloudFront-distribution-ID"}
            }
        }
    ]
}
```

7. [Save] を選択します。

Amazon S3 bucket

Amazon S3 バケットにアクセス許可を追加するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. ナビゲーションペインで、バケットを選択します。
3. 使用する Amazon S3 バケットを選択します。
4. [アクセス許可] タブを選択します。
5. [バケットポリシー] セクションで、[編集] を選択します。
6. 以下のポリシーを [ポリシー] フィールドに貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "*",
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:DataAccessPointAccount": "AWS-account-ID"
                }
            }
        }
    ]
}
```

7. [Save changes] (変更の保存) をクリックします。

ウェブサイトのエンドポイントとして設定された Amazon S3 バケットを使用する

ウェブサイトエンドポイントとして設定されている Amazon S3 バケットを、CloudFront のカスタムオリジンとして使用できます。CloudFront ディストリビューションを設定するときに、オリジンにバケットのエンドポイントをホストしている Amazon S3 の静的ウェブサイトを入力します。この値は、Amazon S3 コンソールの [Static Website Hosting] (静的ウェブサイトホスティング) ペインの [Properties] (プロパティ) タブに表示されます。例:

`http://bucket-name.s3-website-region.amazonaws.com`

Amazon S3 静的ウェブサイトエンドポイントを指定する方法の詳細については、Amazon Simple Storage Service ユーザーガイドの「ウェブサイトエンドポイント」を参照してください。

この形式でバケット名をオリジンとして指定すると、Amazon S3 リダイレクトと Amazon S3 カスタムエラードキュメントを使用できます。詳細については、Amazon Simple Storage Service ユーザーガイドの「カスタムエラードキュメントの設定」および「リダイレクトの設定」を参照してください。(CloudFront もカスタムエラーページを提供します。詳細については、「the section called "HTTP ステータスコード別のカスタムエラーページを作成する"」を参照してください)。

Amazon S3 バケットを CloudFront オリジンサーバーとして使用しても、バケットは変更されません。通常使用しているとおりに使用でき、通常の Amazon S3 料金が発生します。CloudFront の使用料の詳細については、「Amazon CloudFront 料金表」を参照してください。

> **Note**
>
> CloudFront API を使用して、ウェブサイトエンドポイントとして設定された Amazon S3 バケットを使用したディストリビューションを作成する場合、ウェブサイトが Amazon S3 バケットでホストされている場合でも、`CustomOriginConfig` を使用して設定する必要があります。CloudFront API を使用したディストリビューションを作成する方法の詳細については、Amazon CloudFront API リファレンスの「CreateDistribution」を参照してください。

既存の Amazon S3 バケットに CloudFront を追加する

オブジェクトを Amazon S3 バケットに保存している場合は、ユーザーがオブジェクトを S3 から直接取得できるようにするか、または CloudFront が S3 からオブジェクトを取得してからユーザーに配信するように設定できます。使用量が多ければ、Amazon S3 データ転送よりも CloudFront データ転送のほうが料金が安いので、ユーザーがオブジェクトに頻繁にアクセスする場合は、CloudFront

を使用するほうがコストパフォーマンスが向上します。また、Amazon S3 だけからダウンロードするよりも、CloudFront を使用したほうが、ユーザーにより近い場所にオブジェクトが保存されているので、より速くダウンロードできます。

> **Note**
>
> CloudFront で Amazon S3 の Cross-Origin Resource Sharing 設定を尊重する場合は、選択したヘッダーを Amazon S3 に転送するように `Origin` を設定します。詳細については、「the section called “リクエストヘッダーに基づいてコンテンツをキャッシュする”」を参照してください。

現在、Amazon S3 バケットのドメイン名 (DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com など) を使わずに、お客様自身のドメイン名 (example.com など) を使って Amazon S3 バケットから直接コンテンツを配信している場合は、以下の手順を行うことにより、支障なく CloudFront を追加できます。

Amazon S3 からすでにコンテンツを配信しているときに CloudFront を追加するには

1. CloudFront ディストリビューションを作成します。詳細については、「the section called “ディストリビューションを作成する”」を参照してください。

   ディストリビューションを作成するときに、オリジンサーバーとして Amazon S3 バケットの名前を指定します。

   > **Important**
   >
   > バケットが CloudFront で正常に機能するには、その名前が DNS 命名要件に沿ったものでなければなりません。詳細については、Amazon Simple Storage Service ユーザーガイドの「バケットの命名規則」を参照してください。

   Amazon S3 で CNAME を使用している場合、ディストリビューションにもその CNAME を指定します。

2. Amazon S3 バケット内にあるパブリックに読み出し可能なオブジェクトへのリンクが含まれているテストウェブページを作成して、リンクをテストします。この初期テストでは、オブジェクト URL にお客様のディストリビューションの CloudFront ドメイン名 (例: `https://d111111abcdef8.cloudfront.net/images/image.jpg`) を使用します。

CloudFront URL の形式の詳細については、「the section called “ファイルの URL をカスタマイズする”」を参照してください。

3. Amazon S3 の CNAME を使用している場合、アプリケーションでは、バケットの名前 (例: DOC-EXAMPLE-BUCKET.s3.amazonaws.com) を使用する代わりにお客様のドメイン名 (例: example.com) を使用して、Amazon S3 バケットのオブジェクトを参照しています。引き続き、ディストリビューションの CloudFront ドメイン名 (例: d111111abcdef8.cloudfront.net) を使用する代わりにお客様のドメイン名を使用してオブジェクトを参照する場合、DNS サービスプロバイダーを使用して設定を更新する必要があります。

   Amazon S3 の CNAME が機能するためには、DNS サービスプロバイダーに、現在ドメインに対するクエリを Amazon S3 バケットにルーティングしているドメインの CNAME リソースレコードセットが必要です。たとえば、ユーザーが次のオブジェクトをリクエストした場合、

   ```
   https://example.com/images/image.jpg
   ```

   このリクエストは自動的に再ルーティングされ、次のオブジェクトがユーザーに表示されます。

   ```
   https://DOC-EXAMPLE-BUCKET.s3.amazonaws.com/images/image.jpg
   ```

   クエリを Amazon S3 バケットではなく CloudFront ディストリビューションにルーティングするには、DNS サービスプロバイダーから提供された方法を使用して、ドメインの CNAME リソースレコードセットを更新する必要があります。この更新された CNAME レコードによって、お客様のドメインからディストリビューションの CloudFront ドメイン名に、DNS クエリがリダイレクトされます。詳細については、DNS サービスプロバイダーから提供されたドキュメントを参照してください。

   > **Note**
   >
   > Route 53 を DNS サービスとして使用している場合は、CNAME リソースレコードセットまたはエイリアスリソースレコードセットを使用できます。リソースレコードセットの編集については、「レコードの編集」を参照してください。エイリアスリソースレコードセットの詳細については、「エイリアスおよび非エイリアスリソースレコードの選択」を参照してください。どちらのトピックも、Amazon Route 53 開発者ガイドにあります。

   CloudFront での CNAME の使用の詳細については、「the section called “カスタム URL を使用する”」を参照してください。

CNAME リソースレコードセットを更新してから変更が DNS システム全体に伝達されるまで最大で 72 時間かかりますが、通常は、それよりも早く終了します。この間、コンテンツに対する一部の要求は引き続き Amazon S3 バケットにルーティングされ、それ以外の要求は CloudFront にルーティングされます。

Amazon S3 バケットを別の AWS リージョン に移動する

CloudFront ディストリビューションのオリジンとして Amazon S3 を使用していて、バケットを別の AWS リージョンに移動する場合、以下の両方に該当するときは、CloudFront でレコードを更新して新しいリージョンを使用するまでに最大 1 時間かかることがあります。

- CloudFront オリジンアクセスアイデンティティ (OAI) を使用してバケットへのアクセスを制限している
- バケットの移動先の Amazon S3 リージョンで認証に署名バージョン 4 が要求される

OAI を使用している場合、CloudFront はリージョン (数ある値の中で) を使用して、バケットのオブジェクトをリクエストするために使用する署名を計算します。OAI の詳細については、「the section called “オリジンアクセスアイデンティティを使用する (レガシー、非推奨)”」を参照してください。署名バージョン 2 をサポートする AWS リージョンのリストについては、「Amazon Web Services 全般のリファレンス」の「署名バージョン 2 の署名プロセス」を参照してください。

CloudFront レコードの更新を高速化させるには、CloudFront コンソールの [General] (全般) タブの [Descrption] (説明) フィールドを更新するなどして、CloudFront ディストリビューションを更新できます。ディストリビューションを更新すると、CloudFront はバケットがあるリージョンを即座に確認します。すべてのエッジロケーションへの変更の伝達には数分しかかかりません。

MediaStore コンテナまたは MediaPackage チャネルを使用する

CloudFront を使用してビデオをストリーミングするには、まず、MediaStore コンテナとして設定された Amazon S3 バケットを設定するか、MediaPackage でチャネルとエンドポイントを作成します。その後、CloudFront でディストリビューションを作成し設定して、ビデオをストリーミングします。

詳細と手順については、以下のトピックを参照してください。

- the section called “AWS Elemental MediaStore をオリジンとして使用してビデオを配信する”
- the section called “AWS Elemental MediaPackage でフォーマットされたライブ動画を配信する”

Application Load Balancer を使用する

オリジンが 1 つ以上の Amazon EC2 インスタンスでホストされている 1 つ以上の HTTP サーバー (ウェブサーバー) である場合、インターネットに接続している Application Load Balancer を使用して、インスタンスにトラフィックを分散できます。インターネットに接続しているロードバランサーは、パブリックに解決可能な DNS 名を持ち、クライアントからのリクエストをインターネット経由でターゲットにルーティングします。

ロードバランサーに直接アクセスするのではなく CloudFront 経由でのみウェブサーバーにアクセスできるようにする方法など、CloudFront のオリジンとしてApplication Load Balancer を使用する方法の詳細については、「the section called “Application Load Balancer へのアクセスを制限する”」を参照してください。

Lambda 関数 URL を使用する

Lambda 関数 URL は、Lambda 関数専用の HTTPS エンドポイントです。Lambda 関数 URL を使用して、サーバーレスウェブアプリケーションを完全に Lambda 内で構築できます。API Gateway またはApplication Load Balancer と統合する必要なく、関数 URL を介して Lambda ウェブアプリケーションを直接呼び出すことができます。

関数 URL を持つ Lambda 関数を使用することによってサーバーレスウェブアプリケーションを構築する場合、CloudFront を追加して次の利点を得ることができます。

- コンテンツを視聴者の近くにキャッシュしてアプリケーションを高速化する
- ウェブアプリケーションのカスタムドメイン名を使用する
- CloudFront キャッシュ動作を使用して、さまざまな URL パスを異なる Lambda 関数にルーティングする
- CloudFront の地理的制限または AWS WAF (あるいはその両方) を使用して特定のリクエストをブロックする
- CloudFront と共に AWS WAF を使用すると、悪意のあるボットからアプリケーションを保護し、一般的なアプリケーションの悪用を防ぎ、DDoS 攻撃からの保護を強化できます。

CloudFront ディストリビューションのオリジンとして Lambda 関数 URL を使用するには、オリジンドメインとして Lambda 関数 URL の完全なドメイン名を指定します。Lambda 関数 URL ドメイン名は、次の形式を使用します。

function-URL-ID.lambda-url.*AWS-Region*.on.aws

CloudFront ディストリビューションのオリジンとして Lambda 関数 URL を使用する場合は、関数 URL がパブリックにアクセス可能であることを確認する必要があります。これには、次のいずれかの方法を使用します。

- オリジンアクセスコントロール (OAC) を使用する場合は、Lambda 関数 URL の `AuthType` パラメータで `AWS_IAM` 値を使用し、リソースベースのポリシーで `lambda:InvokeFunctionUrl` アクセス許可を許可する必要があります。OAC での Lambda 関数 URL の使用の詳細については、「AWS Lambda 関数 URL オリジンへのアクセスを制限する」を参照してください。
- OAC を使用しない場合は、関数の URL の `AuthType` パラメータを `NONE` に設定し、リソースベースのポリシーで `lambda:InvokeFunctionUrl` 許可を許可できます。

CloudFront がオリジンに送信するリクエストにカスタムオリジンヘッダーを追加し、ヘッダーがリクエストに存在しない場合はエラーレスポンスを返す関数コードを記述することもできます。これにより、ユーザーが Lambda 関数 URL を直接使用するのではなく、CloudFront 経由でのみウェブアプリケーションにアクセスできるようにするのに役立ちます。

Lambda 関数 URL の詳細については、AWS Lambda デベロッパーガイドの以下のトピックを参照してください。

- Lambda 関数 URL – Lambda 関数 URL 機能の一般的な概要
- Lambda 関数 URL の呼び出し – サーバーレスウェブアプリケーションのコーディングに使用するリクエストおよびレスポンスのペイロードに関する詳細が含まれます。
- Lambda 関数 URLs のセキュリティおよび認証モデル — Lambda 認証タイプに関する詳細が含まれます。

Amazon EC2 (または別のカスタムオリジン) を使用する

カスタムオリジンは、クライアントからインターネット経由でターゲットにリクエストをルーティングするパブリックに解決可能な DNS 名を持つ HTTP(S) ウェブサーバーです。HTTP(S) サーバーは、Amazon EC2 インスタンスなど、AWS でホストするか、別の場所でホストできます。ウェブサイトエンドポイントとして設定された Amazon S3 オリジンは、カスタムオリジンと見なされます。詳細については、「the section called “ウェブサイトのエンドポイントとして設定された Amazon S3 バケットを使用する”」を参照してください。

独自の HTTP サーバーをカスタムオリジンとして使用する場合、サーバーの DNS 名に加えて、オリジンからオブジェクトを取得するときに CloudFront が使用する HTTP ポート、HTTPS ポート、およびプロトコルを指定します。

カスタムオリジンを使用する場合、プライベートコンテンツを例外として、ほとんどの CloudFront 機能がサポートされます。署名付き URL を使用してカスタムオリジンからコンテンツを配信できますが、CloudFront の場合、カスタムオリジンにアクセスするには、オリジンがパブリックにアクセス可能な状態になっている必要があります。詳細については、「the section called “署名付き URL と署名付き Cookie を使用したコンテンツを制限する”」を参照してください。

Amazon EC2 インスタンスおよび他のカスタムオリジンを CloudFront とともに使用するには、次のガイドラインに従います。

- 同じ CloudFront オリジンのコンテンツを提供するすべてのサーバーで同じコンテンツをホストし、提供します。詳細については、「the section called “ディストリビューションの設定”」トピックの「the section called “オリジンの設定”」を参照してください。
- AWS Support または CloudFront を使用してこの値をデバッグに使用する必要がある場合は、すべてのサーバー上の `X-Amz-Cf-Id` ヘッダーエントリを口グに記録します。
- カスタムオリジンがリッスンしている HTTP および HTTPS ポートへのリクエストを制限します。
- 実装内のすべてのサーバーの時計を同期します。CloudFront では、署名付き URL と署名付き Cookie、ログ、およびレポートで協定世界時 (UTC) を使用します。さらに、CloudWatch メトリクスを使用して CloudFront アクティビティをモニタリングする場合は、CloudWatch も UTC を使用することに注意してください。
- 冗長サーバーを使用して障害に対処します。
- カスタムオリジンを使用したプライベートコンテンツ供給の詳細については、「the section called “カスタムオリジンのファイルへのアクセスを制限する”」を参照してください。
- リクエストとレスポンス動作、およびサポートされる HTTP ステータスコードについては、「リクエストとレスポンスの動作」を参照してください。

カスタムオリジンで Amazon EC2 を使用する場合は、以下の操作を行うことをお勧めします。

- ウェブサーバーのソフトウェアを自動的にインストールする Amazon マシンイメージを使用します。詳細については、「Amazon EC2 ドキュメント」を参照してください。
- Elastic Load Balancing ロードバランサーを使用して、複数の Amazon EC2 インスタンスにわたるトラフィックを処理するほかに、Amazon EC2 インスタンスの変更からアプリケーションを隔離します。たとえば、ロードバランサーを使用する場合、アプリケーションを変更せずに Amazon

EC2 インスタンスの追加と削除ができます。詳細については、「Elastic Load Balancing ドキュメント」を参照してください。

- CloudFront ディストリビューションを作成する場合は、オリジンサーバーのドメイン名にロードバランサーの URL を指定します。詳細については、「the section called “ディストリビューションを作成する”」を参照してください。

CloudFront オリジングループを使用する

たとえば、高可用性が必要なシナリオでオリジンフェイルオーバーを設定する場合は、CloudFront オリジンのオリジングループを指定できます。オリジンフェイルオーバーを使用して、CloudFront のプライマリオリジンと、プライマリオリジンが特定の HTTP ステータスコードの失敗応答を返したときに CloudFront が自動的に切り替わる 2 番目のオリジンを指定します。

オリジングループをセットアップするステップを含む詳細については、「the section called “オリジンフェイルオーバーを使用して可用性を高める”」を参照してください。

代替ドメイン名 (CNAME) を追加することによって、カスタム URL を使用する

ディストリビューションを作成すると、CloudFront はドメイン名 (d111111abcdef8.cloudfront.net など) を割り当てます。この割り当てられたドメイン名の代わりに、代替ドメイン名 (CNAME とも呼ばれます) を使用できます。

独自のドメイン名 (www.example.com など) を使用する方法については、以下のトピックを参照してください。

トピック

- 代替ドメイン名を使用するための要件
- 代替ドメイン名の使用に対する制限
- 代替ドメイン名を追加する
- 代替ドメイン名を別のディストリビューションに移動する
- 代替ドメイン名を削除する
- 代替ドメイン名でワイルドカードを使用する

代替ドメイン名を使用するための要件

www.example.com などの代替ドメイン名を CloudFront ディストリビューションに追加する場合、次の要件があります。

代替ドメイン名は小文字を使用する必要があります

すべての代替ドメイン名 (CNAME) には小文字を使用する必要があります。

代替ドメイン名は有効な SSL/TLS 証明書の対象であることが必要です

CloudFront ディストリビューションに代替ドメイン名 (CNAME) を追加するには、この代替ドメイン名を対象とする信頼される有効な SSL/TLS 証明書をディストリビューションにアタッチする必要があります。これにより、ドメインの証明書にアクセスできる人物のみがドメインに関連する CNAME に CloudFront を関連付けることができます。

信頼される証明書は、AWS Certificate Manager (ACM) または別の有効な認証局 (CA) から発行されたものです。自己署名証明書は、既存の CNAME の検証には使用できますが、新しい CNAME には使用できません。CloudFront では Mozilla と同じ認証機関をサポートしています。最新のリストは、「Mozilla に付属する CA 証明書一覧」を参照してください。

ワイルドカードを含む代替ドメイン名を含め、アタッチした証明書を使用して代替ドメイン名を確認するために、CloudFront は証明書のサブジェクト代替名 (SAN) を確認します。追加する代替ドメイン名は、SAN の対象である必要があります。

> **Note**
>
> CloudFront ディストリビューションには、一度に 1 つの証明書のみをアタッチすることができます。

ディストリビューションに特定の代替ドメイン名を追加する許可があることを証明するには、次のいずれかを実行します。

- 代替ドメイン名 (product-name.example.com など) を含む証明書を添付します。
- ドメイン名の先頭に * ワイルドカードを含む証明書をアタッチして、1 つの証明書で複数のサブドメインを対象とします。ワイルドカードを指定する場合、複数のサブドメインを CloudFront の代替ドメイン名として追加できます。

次の例では、証明書のドメイン名のワイルドカードを使用して特定の代替ドメイン名を CloudFront に追加することを許可する方法を示しています。

- marketing.example.com を代替ドメイン名として追加するとします。証明書にドメイン名 *.example.com をリストします。この証明書を CloudFront にアタッチすると、この位置でワイルドカードを置き換える一意の代替ドメイン名をディストリビューションに追加することができます (marketing.example.com など)。また、たとえば次の代替ドメイン名を追加することもできます。
  - product.example.com
  - api.example.com

  ただし、ワイルドカードより高いあるいは低い位置に代替ドメイン名を追加することはできません。例えば、代替ドメイン名 example.com や marketing.product.example.com を追加することはできません。
- example.com を代替ドメイン名として追加するとします。これを行うには、ディストリビューションにアタッチした証明書にこのドメイン名 example.com 自体をリストする必要があります。
- marketing.example.com を代替ドメイン名として追加するとします。これを行うには、証明書に *.product.example.com をリストするか、または証明書に marketing.product.example.com 自体をリストできます。

DNS 設定を変更する権限

代替ドメイン名を追加するとき、CNAME レコードを作成して代替ドメイン名の DNS クエリを CloudFront ディストリビューションにルーティングする必要があります。これを行うには、DNS サービスプロバイダーを使用して使用する代替ドメイン名に CNAME を作成する権限があることが必要です。通常、これはドメインを所有していることを指しますが、ドメイン所有者向けにアプリケーションを開発している場合にも当てはまります。

代替ドメイン名と HTTPS

代替ドメイン名を含む HTTPS をビューワーが使用するように構成する場合は、いくつかの追加設定を実行する必要があります。詳細については、「代替ドメイン名と HTTPS を使用する」を参照してください。

代替ドメイン名の使用に対する制限

代替ドメイン名の使用には、以下の制限があることに注意してください。

代替ドメイン名の最大数

ディストリビューションに対して作成できる代替ドメイン名の現在の最大数、またはクォータの引き上げを要求する代替ドメイン名の最大数については、「ディストリビューションの一般的なクォータ」を参照してください。

重複する代替ドメイン名

同じ代替ドメイン名が別の CloudFront ディストリビューションに既に存在する場合は、AWS アカウントが他のディストリビューションを所有しているとしても、その代替ドメイン名を CloudFront ディストリビューションに追加することはできません。

ただし、*.example.com のようなワイルドカード付きの代替ドメイン名を追加できます。これには、www.example.com のような非ワイルドカードの代替ドメイン名を含む (つまり重複している) が含まれます。2 つのディストリビューションで代替ドメイン名が重複している場合、CloudFront は、DNS レコードが指しているディストリビューションに関係なく、より具体的な名前が一致しているディストリビューションにリクエストを送信します。例えば、marketing.domain.com は *.domain.com より具体的です。

ドメインフロンティング

CloudFront には、異なるAWSアカウント間で発生するドメインフロンティングに対する保護が含まれています。ドメインフロンティングとは、非標準クライアントが 1 つの AWS アカウントのドメイン名への TLS/SSL 接続を作成し、別の AWS アカウントの関連しない名前の HTTPS リクエストを行うシナリオです。例えば、TLS 接続が www.example.com に接続を行い、次に www.example.org に HTTP リクエストを送信するような場合です。

異なる AWS アカウント間でドメインフロンティングが発生するケースを防ぐため、CloudFront は、証明書を所有し、CloudFront が特定の接続に関するサービスを提供する AWS アカウントと、リクエストを所有し、CloudFront が同じ接続でそれを処理する AWS アカウントが常に一致することを確実にします。

2 つのAWSアカウント番号が一致しない場合、CloudFront は HTTP 421 Misdirected Request レスポンスを返して、正しいドメインを使用して接続する機会をクライアントに与えます。

ドメインのトップノード (zone apex) の代替ドメイン名の追加

ディストリビューションに代替ドメイン名を追加する場合、通常、CloudFront ディストリビューションにドメイン名の DNS クエリをルーティングするように、DNS 設定で CNAME レコードを作成します。ただし、DNS プロトコルでは、zone apex とも呼ばれる、DNS 名前空間の最上位ノードに対して CNAME レコードを作成することができませ

ん。例えば、example.com という DNS 名を登録する場合、Zone Apex は example.com になります。「example.com」に対して CNAME レコードを作成することはできませんが、www.example.com、newproduct.example.com などに対しては CNAME レコードを作成できます。

DNS サービスとして Route 53 を使用している場合、エイリアスリソースレコードセットを作成できます。これは、CNAME レコードに比べて 2 つの利点があります。トップノードのドメイン名 (example.com) に対してエイリアスリソースレコードセットを作成することもできます。また、エイリアスリソースレコードセットを使用した場合、Route 53 クエリに対する料金はかかりません。

> **Note**
>
> IPv6 を有効にする場合、2 つのエイリアスリソースレコードセットを作成する必要があります。IPv6 トラフィック (A レコード) をルーティングするため、および IPv4 トラフィック (AAAA レコード) をルーティングするためです。詳細については、トピック「ディストリビューション設定リファレンス」の「IPv6 を有効にする」を参照してください。

詳細については、『Amazon Route 53 開発者ガイド』の「ドメイン名を使用したトラフィックの Amazon CloudFront ウェブディストリビューションへのルーティング」を参照してください。

代替ドメイン名を追加する

次のタスクリストでは、CloudFront ドメイン名の代わりに独自のドメイン名をリンクで使用できるように、CloudFront コンソールを使用して代替ドメイン名をディストリビューションに追加する方法を説明しています。CloudFront API を使用したディストリビューションの更新については、「ディストリビューションの設定」を参照してください。

> **Note**
>
> ビューワーで代替ドメイン名を含む HTTPS を使用する場合は、「代替ドメイン名と HTTPS を使用する」を参照してください。

開始する前に: ディストリビューションを更新して代替ドメイン名を追加する前に、以下の操作必ず実行してください。

- ドメイン名を Route 53 または別のドメインレジストラに登録します。
- ドメイン名を対象とする許可された認定権限 (CA) から SSL/TLS 証明書を取得します。証明書をディストリビューションに追加して、ドメインを使用する権限があることを確認します。詳細については、「代替ドメイン名を使用するための要件」を参照してください。

代替ドメイン名を追加する

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 更新するディストリビューションの ID を選択します。
3. [General] タブで、[Edit] を選択します。
4. 以下の値を更新します。

   代替ドメイン名 (CNAME)

   代替ドメイン名を追加します。ドメイン名をコンマで区切るか、新しい行にドメイン名を 1 つずつ入力します。

   SSL 証明書

   次の設定を選択します。

   - HTTPS を使用する – [Custom SSL Certificate (独自 SSL 証明書)] を選択して、リストから証明書を選択します。このリストには、AWS Certificate Manager(ACM)によってプロビジョニングされた証明書、別のCAから購入してACMにアップロードした証明書、および別のCAから購入してIAM証明書ストアにアップロードした証明書が含まれています。

     IAM 証明書ストアに証明書をアップロードする場合で、それがリストに表示されない場合は、「SSL/TLS 証明書をインポートする」の手順を確認して、証明書が正しくアップロードされたことを確認します。

     この設定を選択した場合、オブジェクト URL でのみ代替ドメイン名を使用することをお勧めします (https://www.example.com/logo.jpg)。 ディストリビューションドメイン名 (https://d111111abcdef8.cloudfront.net.cloudfront.net/logo.jpg) を使用する場合、[Clients Supported (サポートされるクライアント)] に選択した値に応じてビューワーは以下のように動作します。

     - [All Clients (すべてのクライアント)]: ビューワーが SNI をサポートしていない場合、CloudFront ドメイン名が TLS/SSL 証明書のドメイン名と一致しないため、警告が表示されます。

- [Only Clients that Support Server Name Indication (SNI) (Server Name Indication (SNI) をサポートするクライアントのみ)]: CloudFront はオブジェクトを返さないでビューワーとの接続を中断します。

Clients Supported (サポートされるクライアント)

次のいずれかのオプションを選択します。

- [All Clients (すべてのクライアント)]: CloudFront が専用の IP アドレスを使用して HTTPS コンテンツを供給します。このオプションを選択した場合、有効になっているディストリビューションに SSL/TLS 証明書を関連付けると、追加料金がかかります。詳細については、「Amazon CloudFront の料金」を参照してください。
- [Only Clients that Support Server Name Indication (SNI) (Server Name Indication (SNI) をサポートしているクライアントのみ) (推奨)]: SNI をサポートしていない旧式のブラウザやクライアントでは、別の方法を使用してコンテンツにアクセスする必要があります。

詳細については、「CloudFront で HTTPS リクエストを処理する方法を選択する」を参照してください。

5. [Yes, Edit (はい、編集します)] を選択します。
6. ディストリビューションの [General (全般)] タブで、[Distribution Status (ディストリビューションのステータス)] が [Deployed (デプロイ済み)] に変わっていることを確認します。ディストリビューションに対する更新がデプロイされる前に代替ドメインの使用を試みた場合、以下のステップで作成するリンクは機能しません。
7. トラフィックをルーティングするための代替ドメイン名 (www.example.com など) の DNS サービスを、ディストリビューションの CloudFront ドメイン名 (d111111abcdef8.cloudfront.net など) に設定します。使用する方法は、ドメインの DNS サービスプロバイダーとして、または別のプロバイダーとして Route 53 を使用しているかどうかによって異なります。

> **Note**
>
> DNS レコードが既に指しているのが、更新中のディストリビューション以外のディストリビューションである場合、DNS を更新した後のみディストリビューションに代替ドメイン名を追加できます。詳細については、「代替ドメイン名の使用に対する制限」を参照してください。

Route 53

エイリアスリソースレコードセットを作成します。エイリアスリソースレコードセットを使用した場合、Route 53 クエリに対する料金はかかりません。また、ルートドメイン名 (example.com) に対してエイリアスリソースレコードセットを作成することもできます。DNS では CNAME の使用が許可されていません。詳細については、『Amazon Route 53 開発者ガイド』の「ドメイン名を使用したトラフィックの Amazon CloudFront ウェブディストリビューションへのルーティング」を参照してください。

別の DNS サービスプロバイダー

DNS サービスプロバイダーが提供する方法を使用して、ドメインの CNAME レコードを追加します。この新しい CNAME レコードによって、DNS クエリが代替ドメイン名 (例: www.example.com) からディストリビューションの CloudFront ドメイン名 (例: d111111abcdef8.cloudfront.net) にリダイレクトされます。詳細については、DNS サービスプロバイダーから提供されたドキュメントを参照してください。

> **Important**
>
> お使いの代替ドメイン名に対応する CNAME レコードが既に存在する場合は、そのレコードを更新するか、ディストリビューションの CloudFront ドメイン名を指す新しいレコードに置き換えてください。

8. dig などの DNS ツールを使用して、前のステップで作成した DNS 設定がディストリビューションのドメイン名を指していることを確認します。

   以下の例は、www.example.com ドメインへの dig リクエストと、応答のうちの関連する部分を示しています。

```
PROMPT> dig www.example.com

; <<> DiG 9.3.3rc2 <<> www.example.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15917
;; flags: qr rd ra; QUERY: 1, ANSWER: 9, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.com.      IN    A
```

```
;; ANSWER SECTION:
www.example.com. 10800 IN CNAME d111111abcdef8.cloudfront.net.
...
```

応答セクションには、www.example.com のクエリを CloudFront ディストリビューションドメイン名 d111111abcdef8.cloudfront.net にルーティングする CNAME レコードが示されます。CNAME の右側の名前が CloudFront ディストリビューションのドメイン名であれば、CNAME レコードは適切に構成されています。その名前が、Amazon S3 バケットのドメイン名などの別の値であれば、CNAME レコードの設定が間違っています。この場合、ステップ 7 に戻り、ディストリビューションのドメイン名を指すように CNAME レコードを修正します。

9. ディストリビューションの CloudFront ドメイン名ではなくお客様のドメイン名を持つ URL にアクセスして、代替ドメイン名をテストします。
10. アプリケーションで、CloudFront ディストリビューションのドメイン名ではなく代替ドメイン名を使用するように、オブジェクトへのリンクを変更します。

代替ドメイン名を別のディストリビューションに移動する

ディストリビューションに代替ドメイン名を追加しようとしても、代替ドメイン名が別のディストリビューションで既に使用されている場合は、CNAMEAlreadyExists エラー (指定した 1 つ以上の CNAME が既に別のリソースに関連付けられています) が発生します。例えば、www.example.com をディストリビューションに追加しようとしても、www.example.com が別のディストリビューションに既に関連付けられている場合は、このエラーが発生します。

その場合、既存の代替ドメイン名を 1 つのディストリビューション (ソースディストリビューション) から別のディストリビューション (ターゲットディストリビューション) に移動できます。プロセスの概要を次のステップに示します。詳細については、概要の各ステップにあるリンクを参照してください。

代替ドメイン名を移動するには

1. ターゲットディストリビューションを設定します。このディストリビューションには、移動する代替ドメイン名を対象とする SSL/TLS 証明書が必要です。詳細については、「ターゲットディストリビューションを設定する」を参照してください。
2. ソースディストリビューションを検索します。AWS Command Line Interface (AWS CLI) を使用して、代替ドメイン名が関連付けられているディストリビューションを検索できます。詳細については、「ソースディストリビューションを検索する」を参照してください。

3. 代替ドメイン名を移動します。これを行う方法は、ソースディストリビューションとターゲットディストリビューションが同じ AWS アカウントにあるかどうかによって異なります。詳細については、「the section called “代替ドメイン名を移動する”」を参照してください。

ターゲットディストリビューションを設定する

代替ドメイン名を移動する前に、ターゲットディストリビューション (代替ドメイン名の移動先のディストリビューション) を設定する必要があります。

ターゲットディストリビューションを設定するには

1. 移動する代替ドメイン名を含む SSL/TLS 証明書を取得します。お持ちでない場合は、AWS Certificate Manager (ACM) にリクエストするか、別の認定権限 (CA) から取得して ACM にインポートします。米国東部 (バージニア北部) (us-east-1) リージョンの証明書をリクエストまたはインポートしていることを確認します。
2. ターゲットディストリビューションを作成していない場合は、新しく作成します。ターゲットディストリビューションの作成の一環として、(前のステップの) 証明書をディストリビューションに関連付けます。詳細については、「ディストリビューションを作成する」を参照してください。

   ターゲットディストリビューションを既に作成している場合は、(前のステップの) 証明書をターゲットディストリビューションに関連付けます。詳細については、「ディストリビューションを更新する」を参照してください。
3. 代替ドメイン名をターゲットディストリビューションのディストリビューションドメイン名に関連付ける DNS TXT レコードを作成します。代替ドメイン名の前にアンダースコア (_) を付けて TXT レコードを作成します。次に、DNS の TXT レコードの例を示します。

   ```
   _www.example.com TXT d111111abcdef8.cloudfront.net
   ```

   CloudFront は、この TXT レコードを使用して、代替ドメイン名の所有権を検証します。

ソースディストリビューションを検索する

1 つのディストリビューションから別のディストリビューションに代替ドメイン名を移動する前に、ソースディストリビューション (代替ドメイン名が現在使用されているディストリビューション) を検索する必要があります。ソースディストリビューションとターゲットディストリビューションの両方の AWS アカウント ID が分かっている場合は、代替ドメイン名の移動方法を決定できます。

代替ドメイン名のソースディストリビューションを検索するには

1. 以下の例に示すように、AWS Command Line Interface (AWS CLI) の CloudFront list-conflicting-aliases コマンドを使用します。*www.example.com* を代替ドメイン名に置き換え、*EDFDVBD6EXAMPLE* を、以前に設定したターゲットディストリビューションの ID に置き換えます。ターゲットディストリビューションと同じ AWS アカウントにある認証情報を使用して、このコマンドを実行します。このコマンドを使用するには、ターゲットディストリビューションに対する `cloudfront:GetDistribution` と `cloudfront:ListConflictingAlias` アクセス許可が必要です。

   ```
   aws cloudfront list-conflicting-aliases --alias www.example.com --distribution-
   id EDFDVBD6EXAMPLE
   ```

   コマンドの出力には、指定されたドメイン名と競合または重複するすべての代替ドメイン名のリストが表示されます。次に例を示します。

   - コマンドに www.example.com を指定すると、コマンドの出力には www.example.com および、重複するワイルドカード付き代替ドメイン名 (*.example.com) が含まれます。
   - コマンドに *.example.com を指定すると、コマンドの出力には、*.example.com と、そのワイルドカードでカバーされる代替ドメイン名 (例えば、www.example.com、test.example.com、dev.example.com など) が含まれます。

   コマンドの出力内の代替ドメイン名ごとに、それが関連付けられているディストリビューションの ID と、ディストリビューションを所有する AWS アカウント ID を確認できます。ディストリビューション ID とアカウント ID は部分的に非表示になっているため、所有しているディストリビューションとアカウントを識別できますが、所有していないディストリビューションとアカウントの情報は保護されます。

2. コマンドの出力から、移動する代替ドメイン名のディストリビューションを検索し、ソースディストリビューションの AWS アカウント ID を書き留めます。ソースディストリビューションのアカウント ID とターゲットディストリビューションを作成したアカウント ID を比較し、これら 2 つのディストリビューションが同じ AWS アカウントにあるかどうかを判断します。これは、代替ドメイン名の移動方法を決定するのに役立ちます。

   代替ドメイン名の移動については、次のトピックを参照してください。

代替ドメイン名を移動する

代替ドメインの移動方法については、状況に応じて、次の方法から選択します。

ソースディストリビューションとターゲットディストリビューションが同じ AWS アカウントにある場合

AWS CLI の associate-alias コマンドを使用して、代替ドメイン名を移動します。この方法は、代替ドメイン名が apex ドメイン (ルートドメインとも呼ばれます。例: example.com) である場合を含めて、同じアカウントでのすべての移動で機能します。詳細については、「the section called “associate-alias を使用して代替ドメイン名を移動する”」を参照してください。

ソースディストリビューションとターゲットディストリビューションが異なる AWS アカウントにある場合

ソースディストリビューションにアクセスでき、代替ドメイン名が apex ドメイン (ルートドメインとも呼ばれます。例: example.com) 以外のもので、その代替ドメイン名と重複するワイルドカードをまだ使用していない場合は、ワイルドカードを使用して代替ドメイン名を移動します。詳細については、「the section called “ワイルドカードを使用して代替ドメイン名を移動する”」を参照してください。

ソースディストリビューションの AWS アカウントにアクセスできない場合は、AWS CLI の associate-alias コマンドを使用して代替ドメイン名を移動します。ソースディストリビューションが無効になっている場合は、代替ドメイン名を移動できます。詳細については、「the section called “associate-alias を使用して代替ドメイン名を移動する”」を参照してください。associate-alias コマンドが機能しない場合は、AWS Support に問い合わせてください。詳細については、「the section called “AWS Support にお問い合わせいただき、代替ドメイン名を移動する”」を参照してください。

associate-alias を使用して代替ドメイン名を移動する

ソースディストリビューションがターゲットディストリビューションと同じ AWS アカウントにある場合、または別のアカウントにあるが無効な場合は、AWS CLI の CloudFront associate-alias コマンドを使用して代替ドメイン名を移動できます。

関連エイリアスを使用して代替ドメイン名を移動するには

1. 以下の例に示すように、AWS CLI を使用して CloudFront の associate-alias コマンドを実行します。*www.example.com* を代替ドメイン名に、*EDFDVBD6EXAMPLE* をターゲットディストリビューションの ID に置き換えます。ターゲットディストリビューションと同じ AWS アカウン

トにある認証情報を使用して、このコマンドを実行します。このコマンドの使用には、次の制約があることに注意してください。

- ターゲットディストリビューションに対する cloudfront:AssociateAlias と cloudfront:UpdateDistribution アクセス許可が必要です。
- ソースディストリビューションとターゲットディストリビューションが同じ AWS アカウントにある場合、ソースディストリビューションに対する cloudfront:UpdateDistribution アクセス許可が必要です。
- ソースディストリビューションとターゲットディストリビューションが異なる AWS アカウントにある場合、ソースディストリビューションを無効にする必要があります。
- ターゲットディストリビューションは、「the section called “ターゲットディストリビューションを設定する”」で説明されているように設定する必要があります。

```
aws cloudfront associate-alias --alias www.example.com --target-distribution-
id EDFDVBD6EXAMPLE
```

このコマンドは、ソースディストリビューションから代替ドメイン名を削除し、ターゲットディストリビューションに追加することで、両方のディストリビューションを更新します。

2. ターゲットディストリビューションが完全にデプロイされたら、代替ドメイン名の DNS レコードがターゲットディストリビューションのディストリビューションドメイン名を参照するように DNS 構成を更新します。

ワイルドカードを使用して代替ドメイン名を移動する

ソースディストリビューションがターゲットディストリビューションと異なる AWS アカウントにあり、ソースディストリビューションが有効になっている場合は、ワイルドカードを使用して代替ドメイン名を移動できます。

> **Note**
>
> ワイルドカードを使用して apex ドメイン (example.com など) を移動することはできません。ソースディストリビューションとターゲットディストリビューションが異なる AWS アカウントにある場合の apex ドメインの移動については、AWS Support にお問い合わせくだ

さい。詳細については、「the section called “AWS Support にお問い合わせいただき、代替ドメイン名を移動する”」を参照してください。

ワイルドカードを使用して代替ドメイン名を移動するには

> **Note**
>
> このプロセスでは、ディストリビューションを複数回更新します。各ディストリビューションが最新の変更を完全にデプロイするまで待ってから次のステップに進みます。

1. ターゲットディストリビューションを更新して、移動する代替ドメイン名に対応したワイルドカード付き代替ドメイン名を追加します。例えば、移動する代替ドメイン名を www.example.com にしている場合は、代替ドメイン名 *.example.com をターゲットディストリビューションに追加します。これを行うには、ターゲットディストリビューションの SSL/TLS 証明書にワイルドカード付きのドメイン名を含める必要があります。詳細については、「the section called “ディストリビューションを更新する”」を参照してください。
2. 代替ドメイン名の DNS 設定を更新して、ターゲットディストリビューションのドメイン名を参照します。例えば、移動する代替ドメイン名が www.example.com の場合、www.example.com の DNS レコードを更新して、トラフィックをターゲットディストリビューションのドメイン名 (例: d111111abcdef8.cloudfront.net) にルーティングします。

   > **Note**
   >
   > DNS 設定を更新した後でも、代替ドメイン名はソースディストリビューションによって引き続き提供されます。これは、代替ドメイン名がソースディストリビューションで現在設定されているためです。

3. ソースディストリビューションを更新して代替ドメイン名を削除します。詳細については、「ディストリビューションを更新する」を参照してください。
4. ターゲットディストリビューションを更新して代替ドメイン名を追加します。詳細については、「ディストリビューションを更新する」を参照してください。
5. dig (または類似の DNS クエリツール) を使用して、代替ドメイン名の DNS レコードの解決先がターゲットディストリビューションのドメイン名になっていることを検証します。
6. (オプション) ターゲットディストリビューションを更新してワイルドカード付き代替ドメイン名を削除します。

AWS Support にお問い合わせいただき、代替ドメイン名を移動する

ソースディストリビューションとターゲットディストリビューションが異なる AWS アカウントにある場合に、ソースディストリビューションの AWS アカウントにアクセスできないか、ソースディストリビューションを無効にできない場合は、AWS Support にお問い合わせいただき代替ドメイン名を移動できます。

AWS Support にお問い合わせいただき、代替ドメイン名を移動するには

1. ターゲットディストリビューションをセットアップします。これには、ターゲットディストリビューションを参照する DNS TXT レコードも含まれます。詳細については、「ターゲットディストリビューションを設定する」を参照してください。
2. AWS Support にお問い合わせいただき、ドメインの所有権を証明し、ドメインを新しい CloudFront ディストリビューションに移動するようにリクエストします。
3. ターゲットディストリビューションが完全にデプロイされたら、代替ドメイン名の DNS レコードがターゲットディストリビューションのディストリビューションドメイン名を参照するように DNS 構成を更新します。

代替ドメイン名を削除する

ドメインあるいはサブドメインから CloudFront ディストリビューション にルーティングするトラフィックを停止する場合、このセクションの手順に従って DNS 設定および CloudFront ディストリビューションを更新します。

ディストリビューションから代替ドメイン名を削除し、同時に DNS 設定を更新することが重要です。これにより、後にこのドメイン名を別の CloudFront ディストリビューションに関連付ける際の問題を回避します。代替ドメイン名が既に 1 つのディストリビューションに関連付けられている場合、別のディストリビューションで設定することはできません。

> **Note**
>
> このディストリビューションから代替ドメイン名を削除して、別のディストリビューションに追加するには、「代替ドメイン名を別のディストリビューションに移動する」の手順に従います。ドメインを削除するこの手順を実行し、続けてこのドメインを別のディストリビューションに追加する場合、CloudFront はエッジロケーションに更新するように伝達されることより、このドメインは一定期間新しいディストリビューションにリンクしません。

ディストリビューションから代替ドメイン名を削除するには

1. まず、ドメインのイントラネットトラフィックを CloudFront ディストリビューションではない別のリソース (Elastic Load Balancing ロードバランサーなど) にルーティングします。または、CloudFront にトラフィックをルーティングする DNS レコードを削除することもできます。

   ドメインの DNS サービスに応じて、次のいずれかを実行します。

   - Route 53 を使用している場合、エイリアスレコードまたは CNAME レコードを更新または削除します。詳細については、「レコードの編集」または「レコードの削除」を参照してください。
   - 別の DNS サービスプロバイダーを使用している場合、その DNS サービスプロバイダーが提供するメソッドで CloudFront にトラフィックを送る CNAME レコードを更新または削除します。詳細については、DNS サービスプロバイダーから提供されたドキュメントを参照してください。
2. ドメインの DNS レコードを更新したら、この変更が伝達され、DNS リゾルバーが新しいリソースにトラフィックをルーティングするまで待ちます。URL でドメインを使用するテストリンクを作成することで、以上が完了したことを確認できます。
3. AWS Management Console にサインインして、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開き、次の手順で CloudFront ディストリビューションを更新してドメイン名を削除します。

   a. 更新するディストリビューションの ID を選択します。

   b. [General] タブで、[Edit] を選択します。

   c. [Alternate Domain Names (CNAMEs) (代替ドメイン名 (CNAME))] で、ディストリビューションに使用しない代替ドメイン名 (1 つ以上) を削除します。

   d. [Yes, Edit (はい、編集します)] を選択します。

代替ドメイン名でワイルドカードを使用する

代替ドメイン名を追加するとき、サブドメインを個別に追加する代わりに、ドメイン名の最初に * ワイルドカードを使用できます。例えば、代替ドメイン名を *.example.com にしている場合は、「www.example.com」、「product-name.example.com」、「marketing.product-name.example.com」などの、末尾が「example.com」であるドメイン名であればどれでも URL に使用できます。オブジェクトのパスは、ドメイン名に関係なく同じです。例えば、次のようになります。

- www.example.com/images/image.jpg
- product-name.example.com/images/image.jpg
- marketing.product-name.example.com/images/image.jpg

ワイルドカードを含む代替ドメイン名については、以下の要件に従ってください。

- 代替ドメイン名は、アスタリスクとドット (*.) で始まる必要があります。
- *domain.example.com のように、サブドメイン名の一部をワイルドカードで置き換えることはできません。
- 「subdomain.*.example.com」 のように、ドメイン名の途中にあるサブドメインを置き換えることはできません。
- ワイルドカードを使用する代替ドメイン名を含むすべての代替ドメイン名は、証明書のサブジェクト代替名 (SAN) でカバーされている必要があります。

ワイルドカード付き代替ドメイン名 (*.example.com など) には、使用されている別の代替ドメイン名 (example.com など) を含めることができます。

CloudFront ディストリビューションで WebSockets を使用する

Amazon CloudFront は、WebSocket の使用をサポートしています。これは、クライアントとサーバー間の長時間の双方向性接続が必要な場合に便利な TCP ベースのプロトコルです。永続的な接続は、多くの場合、リアルタイムアプリケーションでの要件です。WebSockets を使用するシナリオには、ソーシャルチャットプラットフォーム、オンラインコラボレーションワークスペース、マルチプレイヤーゲーム、および金融取引プラットフォームのようなリアルタイムのデータフィードを提供するサービスが含まれます。WebSocket 接続経由のデータは、全二重通信に対して双方向に流れることができます。

WebSocket 機能は自動的に有効になり、どのディストリビューションでも動作します。WebSockets を使用するには、ディストリビューションにアタッチされているキャッシュビヘイビアで次のいずれかを設定します。

- すべてのビューワーリクエストのヘッダーをオリジンに転送します (AllViewer マネージドオリジンリクエストポリシーを使用できます)。
- 具体的には、オリジンリクエストポリシーで `Sec-WebSocket-Key` および `Sec-WebSocket-Version` リクエストヘッダーを転送します。

WebSocket プロトコルの仕組み

WebSocket プロトコルは、独立した TCP ベースのプロトコルであり、これによって HTTP のオーバーヘッド (および潜在的なレイテンシーの増加) の一部を回避することができます。

WebSocket 接続を確立するために、クライアントはプロトコルの変更のため HTTP のアップグレードセマンティクスを使用する通常の HTTP リクエストを送信します。その後、サーバーはハンドシェイクを完了できます。WebSocket 接続は開いたままで、クライアントまたはサーバーが毎回新しい接続を確立する必要なしに互いにデータフレームを送信できます。

デフォルトでは、WebSocket プロトコルは通常の WebSocket 接続にポート 80 を使用し、TLS/SSL 経由の WebSocket 接続ではポート 443 を使用します。CloudFront ビューワープロトコルポリシー および プロトコル (カスタムオリジンのみ) に選択したオプションは、HTTP トラフィックだけでなく、WebSocket 接続にも適用されます。

WebSocket の要件

WebSocket リクエストは、以下の標準形式で RFC 6455 に準拠する必要があります。

サンプルクライアントリクエスト:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: https://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

サンプルサーバー応答:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

WebSocket 接続がクライアントまたはサーバーによって、またはネットワークの中断によって切断された場合、クライアントアプリケーションはサーバーとの接続を再開する必要があります。

推奨される WebSocket ヘッダー

WebSockets を使用する際に予期しない圧縮関連の問題を避けるため、オリジンリクエストポリシーに以下のヘッダーを含めることをお勧めします。

- `Sec-WebSocket-Key`
- `Sec-WebSocket-Version`
- `Sec-WebSocket-Protocol`
- `Sec-WebSocket-Accept`
- `Sec-WebSocket-Extensions`

キャッシュと可用性

CloudFront を使用して、オリジンサーバーが直接応答する必要があるリクエストの数を減らすことができます。CloudFront キャッシュを使用すると、ユーザーに近い CloudFront エッジロケーションからより多くのオブジェクトが提供されます。これにより、オリジンサーバーの負荷が軽減され、レイテンシーが減少します。

エッジキャッシュから CloudFront が提供できるリクエスト数が多いほど、オブジェクトの最新バージョンまたは固有バージョンを取得するために CloudFront がオリジンに転送する必要のあるビューワーリクエストが少なくなります。CloudFront を最適化して、オリジンへのリクエストをできるだけ少なくするには、CloudFront Origin Shield の使用を検討してください。詳細については、「Amazon CloudFront Origin Shield の使用」を参照してください。

すべてのリクエスト数に対する、CloudFront キャッシュから直接提供されるリクエストの比率をキャッシュヒット率と呼びます。ヒット、ミス、またはエラーであるビューワーリクエストの割合は CloudFront コンソールで確認できます。詳細については、「CloudFront キャッシュ統計レポートを表示する」を参照してください。

キャッシュヒット率に影響を与えるいくつかの要因があります。以下の「CloudFront キャッシュから直接提供するリクエストの割合 (キャッシュヒット率) を増やす」のガイダンスを参照して、キャッシュヒット率を向上させるために、CloudFront ディストリビューション設定を調整できます。

CloudFront が配信するコンテンツの追加および削除については、「CloudFront が配信するコンテンツを追加、削除、または置き換える」を参照してください。

トピック

- CloudFront キャッシュから直接提供するリクエストの割合 (キャッシュヒット率) を増やす
- Amazon CloudFront Origin Shield の使用
- CloudFront オリジンフェイルオーバーを使用して高可用性を最適化する
- コンテンツをキャッシュに保持する期間 (有効期限) を管理する
- クエリ文字列パラメータに基づいてコンテンツをキャッシュする
- Cookie に基づいてコンテンツをキャッシュする
- リクエストヘッダーに基づいてコンテンツをキャッシュする

CloudFront キャッシュから直接提供するリクエストの割合 (キャッシュヒット率) を増やす

コンテンツのオリジンサーバーにアクセスするのではなく、CloudFront キャッシュから直接提供されるビューワーリクエストの比率を増やしてパフォーマンスを向上させることができます。これは、キャッシュヒット率の向上として知られています。

以下のセクションでは、キャッシュヒット率を改善する方法について説明します。

トピック

CloudFront がオブジェクトをキャッシュする期間を指定する

キャッシュヒット率を向上させるには、Cache-Control max-age ディレクティブをオブジェクトに追加し、`max-age` に対して最も長い実用的な値を指定するようにオリジンを設定します。キャッシュ期間が短ければ短いほど、オブジェクトが変更されているかどうかを特定して最新バージョンを取得するリクエストを CloudFront がオリジンに送信する頻度が高くなります。`stale-while-revalidate` および `stale-if-error` ディレクティブで `max-age` を補足すると、特定の条件下でのキャッシュヒット率をさらに向上させることができます。詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

オリジンシールドを使用する

CloudFront Origin Shield は、オリジンの前に追加のキャッシュレイヤーを提供するため、CloudFront ディストリビューションのキャッシュヒット率を向上させるために役立ちます。Origin Shield を使用すると、CloudFront のすべてのキャッシュレイヤーからオリジンへのすべてのリクエストは、1 つの場所から送信されます。CloudFront は、各オブジェクトを Origin Shield からオリジンへの 1 つのリクエストを使用して取得できます。CloudFront キャッシュの他のすべて

のレイヤー (エッジロケーションとリージョン別エッジキャッシュ) は、OriginShield からオブジェクトを取得できます。

詳細については、「Amazon CloudFront Origin Shield の使用」を参照してください。

クエリ文字列パラメータに基づくキャッシュ

クエリ文字列パラメータに基づいてキャッシュするように CloudFront を設定する場合、以下を行うことでキャッシュを改善できます。

- オリジンが一意のオブジェクトを返すクエリ文字列パラメータのみを転送するように CloudFront を設定する。
- 同じパラメータのすべてのインスタンスで大文字と小文字の区別を統一する。例えば、1 つのリクエストに `parameter1=A` が含まれており、別のリクエストに `parameter1=a` が含まれている場合、CloudFront は `parameter1=A` が含まれているリクエストと `parameter1=a` が含まれているリクエストを 2 つの個別のリクエストとしてオリジンに転送します。これが行われると、オブジェクトは同一であっても、CloudFront はオリジンから個別に返された対応するオブジェクトを個別にキャッシュします。`A` または `a` のどちらかのみを使用すると、CloudFront がオリジンに転送するリクエストが少なくなります。
- パラメータの順序を統一する。大文字と小文字が区別されることと同じように、あるオブジェクトに対するリクエストに `parameter1=a&parameter2=b` というクエリ文字列が含まれており、同じオブジェクトに対する別のリクエストに `parameter2=b&parameter1=a` が含まれている場合、オブジェクトは同一であっても、CloudFront は両方のリクエストをオリジンに転送し、対応するオブジェクトを個別にキャッシュします。パラメータの順序を統一すると、CloudFront がオリジンに転送するリクエストが少なくなります。

詳細については、「クエリ文字列パラメータに基づいてコンテンツをキャッシュする」を参照してください。CloudFront がオリジンに転送するクエリ文字列を確認するには、CloudFront ログファイルの `cs-uri-query` 列の値を参照します。詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

Cookie 値に基づくキャッシュ

Cookie 値に基づいてキャッシュするように CloudFront を設定する場合、以下を行うことでキャッシュを改善できます。

- すべての Cookie を転送する代わりに特定の Cookie のみを転送するように CloudFront を設定する。Cookie をオリジンに転送するように CloudFront を設定する場合、CloudFront は Cookie の名

前と値のすべての組み合わせを転送します。その場合、オリジンが返すオブジェクトがすべて同一であっても、別々にキャッシュされます。

たとえば、ビューワーがすべてのリクエストに 2 つの Cookie を含め、それぞれの Cookie に使用できる値が 3 つあり、Cookie 値のすべての組み合わせが可能であるとします。CloudFront は、各オブジェクトに対して最大 6 つの異なるリクエストをオリジンに転送します。オリジンが 1 つの Cookie のみに基づいて同じオブジェクトの複数のバージョンを返す場合、CloudFront は必要以上のリクエストをオリジンに転送し、同一オブジェクトの複数のバージョンを不必要にキャッシュします。

- 静的コンテンツと動的コンテンツに対してそれぞれ異なるキャッシュ動作を作成し、動的コンテンツの場合にのみ Cookie をオリジンに転送するように CloudFront を設定する。

  例えば、ディストリビューションのキャッシュ動作が 1 つしかなく、このディストリビューションを `.js` ファイルなどの動的コンテンツと頻繁に変更されない `.css` ファイルの両方に使用するとします。CloudFront は Cookie 値に基づいて個別のバージョンの `.css` ファイルをキャッシュするため、それぞれの CloudFront エッジロケーションがすべての新しい Cookie 値または Cookie 値の組み合わせに対してリクエストをオリジンに転送します。

  Cookie 値に基づいて CloudFront がキャッシュしない、`*.css` というパスパターンのキャッシュ動作を作成すると、CloudFront は、エッジロケーションが特定の `.css` ファイルに対して受け取る最初のリクエストおよび `.css` ファイルの有効期限が切れた後の最初のリクエストでのみ `.css` ファイルのリクエストをオリジンに転送します。
- Cookie 値がユーザーごとに一意である (ユーザー ID など) 動的コンテンツと、より少ない数の一意の値に基づいて変化する動的コンテンツに対してそれぞれ異なるキャッシュ動作を作成する (可能な場合)。

詳細については、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。CloudFront がオリジンに転送する Cookie を確認するには、CloudFront ログファイルの `cs(Cookie)` 列の値を参照します。詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

リクエストヘッダーに基づくキャッシュ

リクエストヘッダーに基づいてキャッシュするように CloudFront を設定する場合、以下を行うことでキャッシュを改善できます。

- すべてのヘッダーに基づいて転送およびキャッシュする代わりに特定のヘッダーのみに基づいて転送およびキャッシュするように CloudFront を設定します。CloudFront は、指定したヘッダーの名

前と値のすべての組み合わせを転送します。その場合、オリジンが返すオブジェクトがすべて同一であっても、別々にキャッシュされます。

> **Note**
>
> CloudFront は常に以下のトピックで指定されているヘッダーをオリジンに転送します。
>
> - CloudFront がリクエストを処理して Amazon S3 オリジンサーバーに転送する方法 > CloudFront が削除または更新する HTTP リクエストヘッダー
> - CloudFront がリクエストを処理してカスタムオリジンサーバーに転送する方法 > HTTP リクエストヘッダーと CloudFront の動作 (カスタムオリジンおよび Amazon S3 オリジン)

リクエストヘッダーに基づいてキャッシュするように CloudFront を設定する場合、CloudFront が転送するヘッダーを変更するのではなく、CloudFront がヘッダー値に基づいてオブジェクトをキャッシュするかどうかのみを変更します。

- 多数の一意の値を持つリクエストヘッダーに基づいてキャッシュすることを可能な限り回避する。

  例えば、ユーザーのデバイスに基づいてさまざまなサイズのイメージを提供する場合、使用できる値が多数ある `User-Agent` ヘッダーに基づいてキャッシュするように CloudFront を設定しないでください。代わりに、CloudFront デバイスタイプヘッダー `CloudFront-Is-Desktop-Viewer`、`CloudFront-Is-Mobile-Viewer`、`CloudFront-Is-SmartTV-Viewer`、`CloudFront-Is-Tablet-Viewer` に基づいてキャッシュするように CloudFront を設定します。さらに、タブレットとデスクトップに同じバージョンのイメージを返す場合は、`CloudFront-Is-Tablet-Viewer` ヘッダーではなく `CloudFront-Is-Desktop-Viewer` ヘッダーのみを転送します。

詳細については、「リクエストヘッダーに基づいてコンテンツをキャッシュする」を参照してください。

圧縮が不要な場合に `Accept-Encoding` ヘッダーを削除する

圧縮が有効になっていない場合 (オリジンがサポートしていないため、CloudFront がサポートしていないため、またはコンテンツが圧縮可能でないため)、Custom Origin Header を以下のように設定するオリジンにディストリビューション内のキャッシュ動作を関連付けることで、キャッシュヒット率を増やすことができます。

- ヘッダー名: Accept-Encoding
- ヘッダー値: (空白のままにする)

この設定を使用すると、CloudFront はキャッシュキーから Accept-Encoding ヘッダーを削除し、オリジンリクエストにヘッダーを含めません。この設定は、そのオリジンからのディストリビューションに CloudFront が配信するすべてのコンテンツに適用されます。

HTTP 経由でメディアコンテンツを提供する

ビデオオンデマンド (VOD) およびビデオコンテンツのストリーミングの最適化の詳細については、「CloudFront を使用したビデオオンデマンドおよびライブストリーミングビデオ」を参照してください。

Amazon CloudFront Origin Shield の使用

CloudFront Origin Shield は、オリジンの負荷を最小限に抑え、可用性を向上させ、運用コストを削減するために役立つ、CloudFront キャッシュインフラストラクチャ内の追加レイヤーです。CloudFront Origin Shield の使用には、次のメリットがあります。

キャッシュヒット率の向上

Origin Shield は、オリジンの前に追加のキャッシュレイヤーを提供するため、CloudFront ディストリビューションのキャッシュヒット率を向上させるために役立ちます。Origin Shield を使用すると、CloudFront のすべてのキャッシュレイヤーからオリジンへのすべてのリクエストが Origin Shield を通過し、キャッシュがヒットする可能性が高くなります。CloudFront は、Origin Shield からの 1 つのオリジンリクエストを使用して各オブジェクトを取得できます。CloudFront キャッシュのその他のレイヤー (エッジロケーションとリージョン別エッジキャッシュ) はすべて、Origin Shield からオブジェクトを取得できます。

オリジンの負荷を軽減

Origin Shield を使用すると、オリジンに送信される同じオブジェクトへの同時リクエストの数をさらに減らすことができます。Origin Shield のキャッシュにないコンテンツへのリクエストは、同じオブジェクトに対する他のリクエストと統合され、オリジンに送信されるリクエストは 1 件のみになります。オリジンで処理するリクエスト数を減らすと、ピークロード時や想定外のトラフィック急増時にオリジンの可用性が維持され、ジャストインタイムパッケージング、画像変換、データ転送 (DTO) などのコストを削減できます。

ネットワークパフォーマンスの向上

オリジンに対するレイテンシーが最も低い AWS リージョンで Origin Shield を有効にすることで、最良のネットワークパフォーマンスが得られます。AWS リージョン内のオリジンの場合、CloudFront のネットワークトラフィックは、オリジンに到達するまで高スループットの CloudFront ネットワーク上に留まります。AWS 外のオリジンの場合、CloudFront のネットワークトラフィックは、オリジンへの低レイテンシー接続がある Origin Shield に到達するまで CloudFront ネットワーク上に留まります。

Origin Shield の使用には追加料金が発生します。詳細については、「CloudFront 料金表」を参照してください。

トピック

- Origin Shield のユースケース
- Origin Shield の AWS リージョンの選択
- Origin Shield の有効化
- Origin Shield のコストの見積もり
- Origin Shield の高可用性
- Origin Shield と他の CloudFront 機能との連携

Origin Shield のユースケース

CloudFront Origin Shield は、次のような多くのユースケースで役立ちます。

- 異なる地理的リージョンにビューワーが分散している場合
- ライブストリーミングまたはオンザフライ画像処理のために、オリジンがジャストインタイムパッケージを提供する場合
- オンプレミスのオリジンに、容量または帯域幅の制約がある場合
- ワークロードが複数のコンテンツ配信ネットワーク (CDN) を使用する場合

Origin Shield は、動的コンテンツがオリジンにプロキシ化される場合、コンテンツのキャッシュ可能性が低い安倍、コンテンツのリクエスト頻度の低い場合など、上記以外の状況には適さないことがあります。

以下のセクションでは、以下のユースケースにおける Origin Shield の利点について説明します。

ユースケース

- 異なる地理的リージョンにビューワーが分散している場合
- 複数の CDN を使用する場合

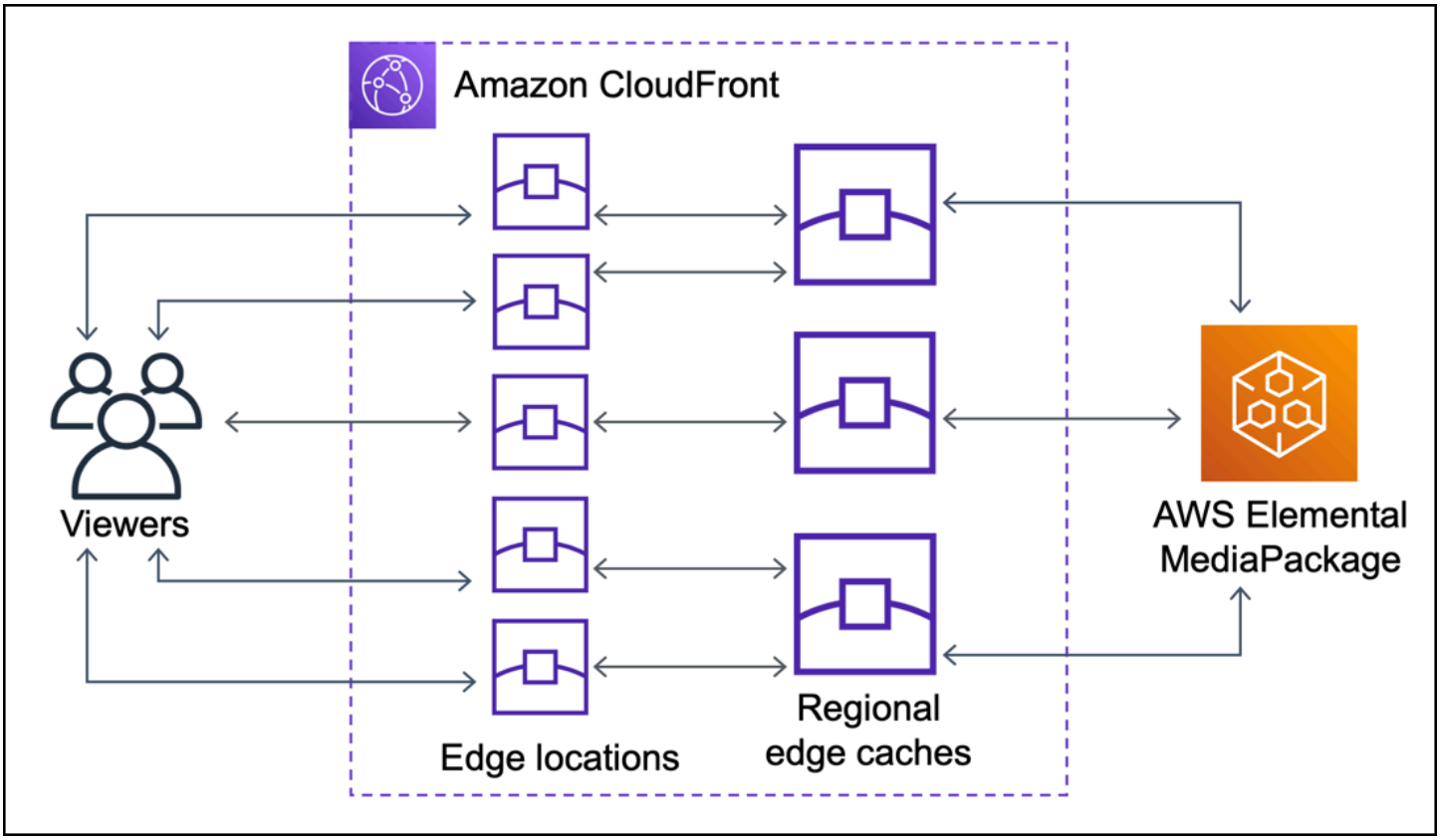
異なる地理的リージョンにビューワーが分散している場合

Amazon CloudFront を使用すると、CloudFront がキャッシュを使用して処理できるリクエストはオリジンに送信されないため、オリジンの負荷が軽減されます。CloudFront が提供するエッジロケーションのグローバルネットワークに加え、リージョン別エッジキャッシュは中間層のキャッシュレイヤーとして機能します。これにより、地理的に近いリージョンのビューワーにキャッシュヒットが提供され、オリジンリクエストが統合されます。ビューワーリクエストは、まず近くの CloudFront エッジロケーションにルーティングされ、オブジェクトがその口ケーションでキャッシュされていない場合、リクエストはリージョン別エッジキャッシュに送信されます。

ビューワーのリージョンが地理的に異なる場合、リクエストは異なるリージョン別エッジキャッシュを介してルーティングされ、それぞれから同じコンテンツに対するリクエストがオリジンに送信される可能性があります。Origin Shield を使用すると、リージョン別エッジキャッシュとオリジンの間にキャッシュのレイヤーが追加されます。すべてのリージョン別エッジキャッシュからのリクエストはすべて、Origin Shield を通過し、オリジンの負荷をさらに軽減します。次の図にその概念を示します。次の図で、オリジンは AWS Elemental MediaPackage です。

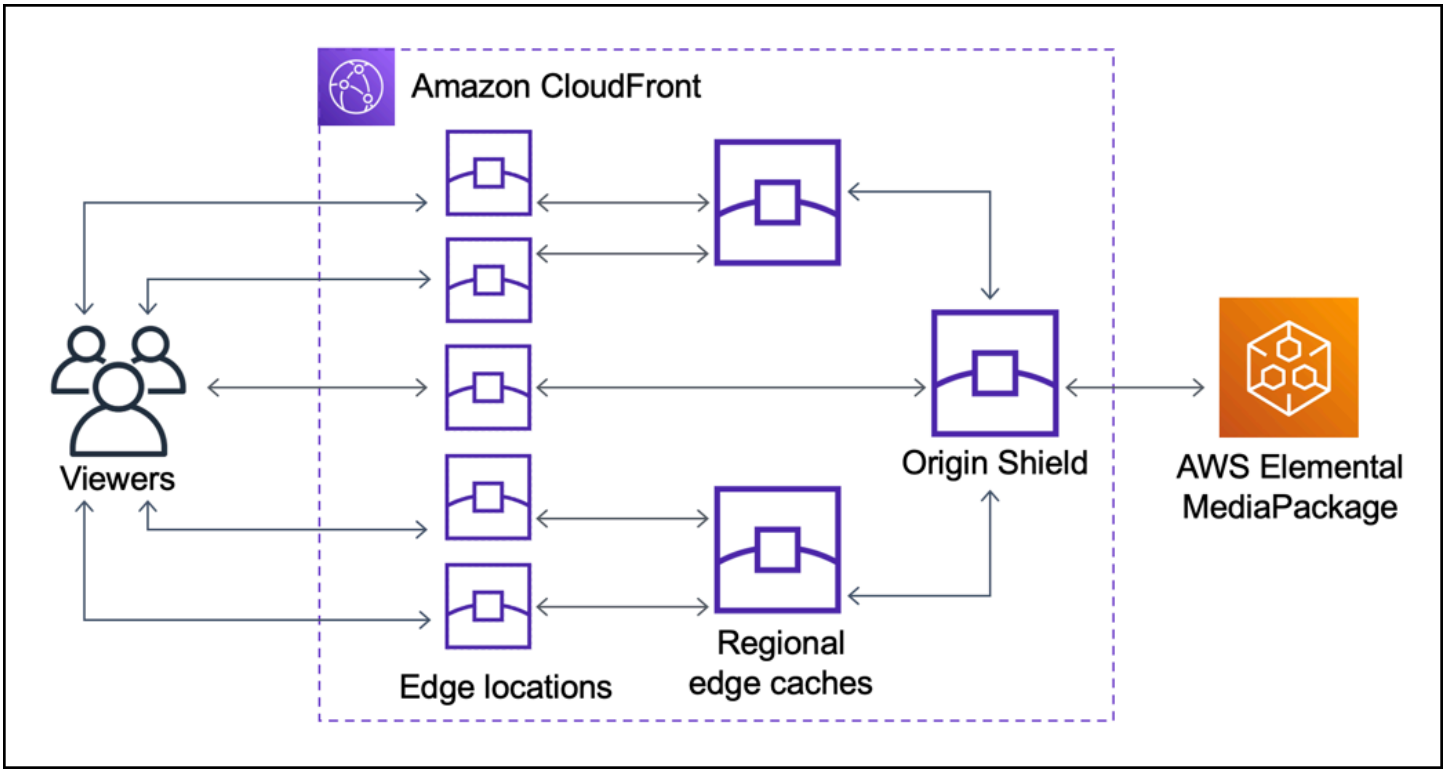
Origin Shield を使用しない場合

Origin Shield を使用しない場合は、次の図に示されているように、同じコンテンツに対するリクエストをオリジンが重複して受け取る可能性があります。



Origin Shield を使用する場合

Origin Shield を使用すると、次の図に示されているように、オリジンの負荷を軽減できます。



複数の CDN を使用する場合

ライブビデオイベントや人気のあるオンデマンドコンテンツを配信するには、複数のコンテンツ配信ネットワーク (CDN) を使用することがあります。複数の CDN の使用には利点もありますが、同じコンテンツに対して多数の重複リクエストをオリジンが受け取る可能性があります。それぞれのリクエストは、異なる CDN から送信されることも、同じ CDN 内の異なる場所から送信されることもあります。このような冗長リクエストにより、オリジンの可用性に影響することも、ジャストインタイムパッケージングやインターネットへのデータ転送 (DTO) などのプロセスに追加の運用コストが発生することもあります。

Origin Shield を使用し、他の CDN のオリジンとして CloudFront ディストリビューションを組み合わせると、次のようなメリットがあります。

- オリジンで受信される冗長リクエストが少なくなるため、複数の CDN を使用した場合の悪影響を軽減できます。
- CDN 全体で共通のキャッシュキーを使用し、オリジン向けの機能を一元管理できます。
- ネットワークパフォーマンスの向上。他の CDN からのネットワークトラフィックが近くの CloudFront エッジロケーションで終了し、ローカルキャッシュからのヒットが発生する可能性があります。リクエストされたオブジェクトがエッジロケーションキャッシュ内にない場合、オリジンへのリクエストは Origin Shield までの間 CloudFront ネットワーク上に残り、オリジンへの高い

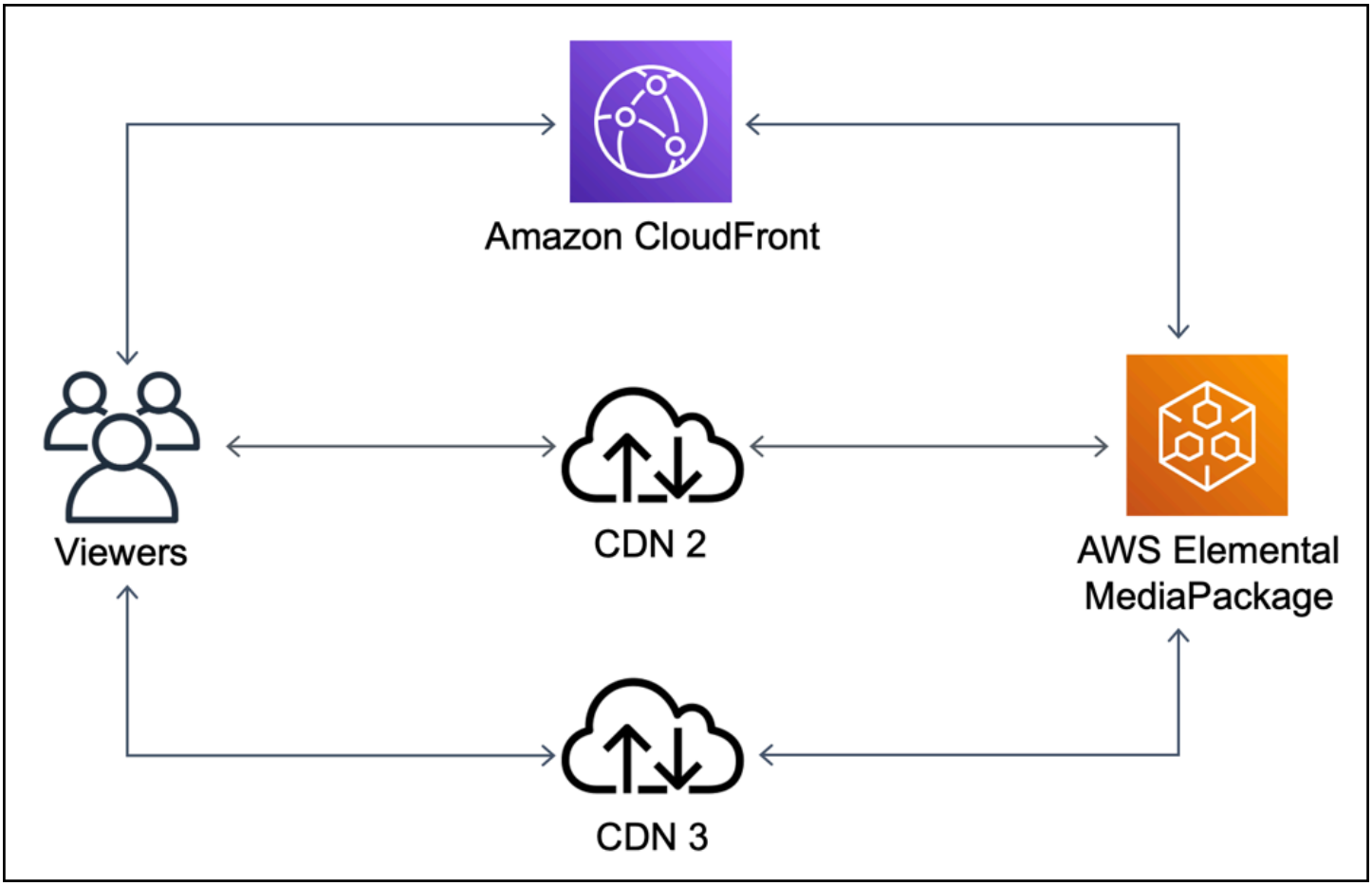
スループットと低レイテンシーを提供します。リクエストされたオブジェクトが Origin Shield のキャッシュ内にある場合、オリジンへのリクエストは完全に回避されます。

> **Important**
>
> マルチ CDN アーキテクチャでオリジンシールドを使用し、割引料金を利用する場合は、[お問い合わせ] または AWS 営業担当者を通じて詳細をご確認ください。追加料金が適用される場合があります。

以下の図は、複数の CDN で人気のあるライブビデオイベントを提供する場合に、この設定がオリジンへの負荷を最小限に抑える方法を示しています。次の図で、オリジンは AWS Elemental MediaPackage です。

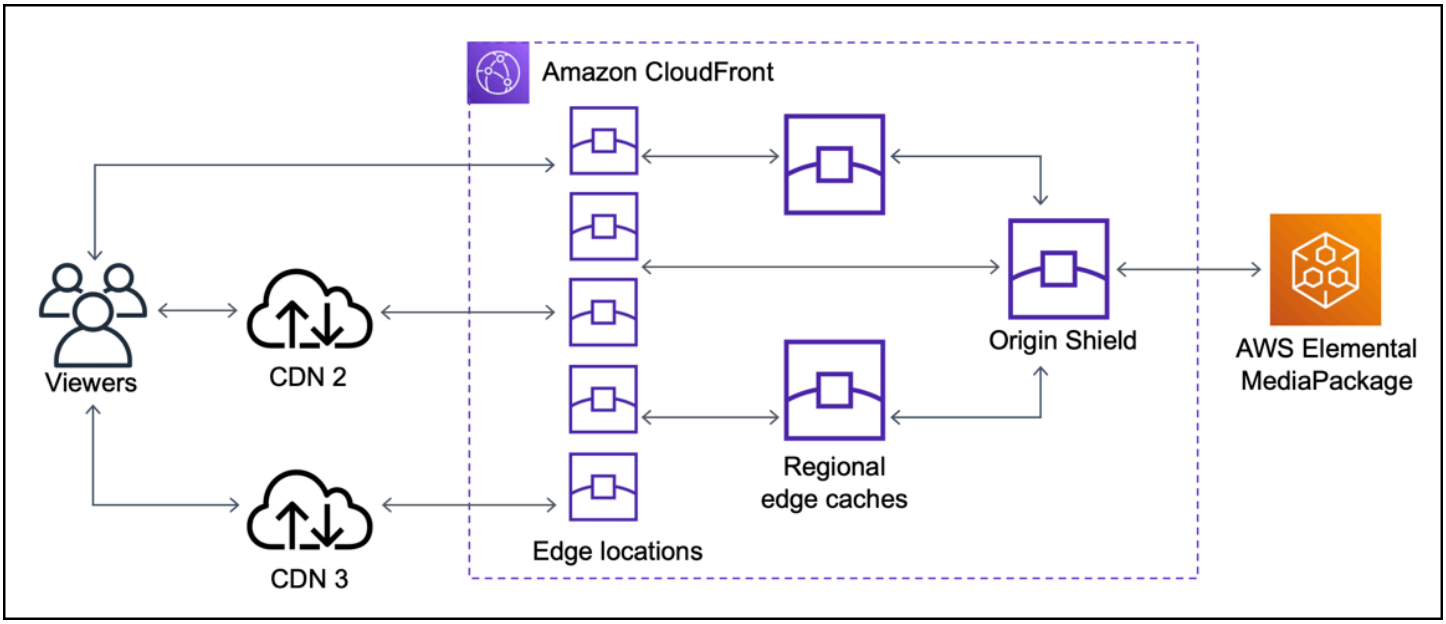
Origin Shield を使用しない場合 (複数の CDN を使用)

Origin Shield を使用しない場合は、次の図に示されているように、同じコンテンツに対する多数の重複リクエストを (それぞれ異なる CDN から) オリジンが受け取る可能性があります。



Origin Shield を使用する場合 (複数の CDN を使用)

Origin Shield を使用し、他の CDN のオリジンとして CloudFront を組み合わせると、次の図に示されているように、オリジンの負荷を軽減できます。



Origin Shield の AWS リージョンの選択

Amazon CloudFront は、CloudFront のリージョンエッジキャッシュがある AWS リージョンで Origin Shield を提供しています。Origin Shield を有効にするときは、Origin Shield の AWS リージョンを選択します。オリジンに対するレイテンシーが最も低い AWS リージョンを選択するようにしてください。Origin Shieldは、AWS リージョン内にあるオリジンにも、AWS 外のオリジンにも使用できます。

AWS リージョン内のオリジン

オリジンが AWS リージョンにある場合は、CloudFront が Origin Shield を提供するリージョン内にオリジンがあるかどうかを最初に判断します。CloudFront は、以下の AWS リージョンで Origin Shield を提供しています。

- 米国東部 (オハイオ) – `us-east-2`
- 米国東部 (バージニア北部) – `us-east-1`
- 米国西部 (オレゴン) – `us-west-2`
- アジアパシフィック (ムンバイ) – `ap-south-1`
- アジアパシフィック (ソウル) – `ap-northeast-2`
- アジアパシフィック (シンガポール) – `ap-southeast-1`
- アジアパシフィック (シドニー) – `ap-southeast-2`
- アジアパシフィック (東京) – `ap-northeast-1`

- 欧州 (フランクフルト) – eu-central-1
- 欧州 (アイルランド) – eu-west-1
- 欧州 (ロンドン) – eu-west-2
- 南米 (サンパウロ) – sa-east-1

CloudFront で Origin Shield が提供されている AWS リージョン内にオリジンがある場合

CloudFront が Origin Shield を提供する AWS リージョン (上記のリストを参照) にオリジンがある場合は、オリジンと同じリージョンで Origin Shield を有効にします。

CloudFront が Origin Shield を提供する AWS リージョン内にオリジンがない場合

CloudFront が Origin Shield を提供する AWS リージョンにオリジンがない場合は、以下の表を参照して、Origin Shield を有効にするリージョンを判断します。

オリジンの場所	Origin Shield を有効にするリージョン
米国西部 (北カリフォルニア) – us-west-1	米国西部 (オレゴン) – us-west-2
アフリカ (ケープタウン) – af-south-1	欧州 (アイルランド) – eu-west-1
アジアパシフィック (香港) – ap-east-1	アジアパシフィック (シンガポール) – ap-southeast-1
カナダ (中部) – ca-central-1	米国東部 (バージニア北部) – us-east-1
欧州 (ミラノ) – eu-south-1	欧州 (フランクフルト) – eu-central-1
欧州 (パリ) – eu-west-3	欧州 (ロンドン) – eu-west-2
欧州 (ストックホルム) – eu-north-1	欧州 (ロンドン) – eu-west-2
中東 (バーレーン) – me-south-1	アジアパシフィック (ムンバイ) – ap-south-1

オリジンが 外にある場合AWS

Origin Shield は、オンプレミスのオリジン、または AWS リージョン外のオリジンにも使用できます。その場合は、オリジンに対するレイテンシーが最も低い AWS リージョンで Origin Shield を有

効にします。オリジンに対するレイテンシーが最も低い AWS リージョンがわからない場合は、以下の提案を参考にして判断することができます。

- 上記の表を参照し、オリジンの地理的位置に基づいて、オリジンに対するレイテンシーが最も低いと思われる AWS リージョンを見計らいます。
- オリジンに地理的に近いいくつかの異なる AWS リージョンで Amazon EC2 インスタンスを起動し、`ping` を使用してテストを数回実行して、これらのリージョンとオリジンの間の典型的なネットワークレイテンシーを測定できます。

Origin Shield の有効化

Origin Shield を有効にすると、キャッシュヒット率の向上、オリジンへの負荷の軽減、パフォーマンスの強化を図ることができます。オリジンシールドを有効にするには、CloudFront ディストリビューションのオリジン設定を変更します。Origin Shield は、オリジンのプロパティです。CloudFront ディストリビューションのオリジンごとに、そのオリジンに最適なパフォーマンスを提供する AWS リージョンで Origin Shield を個別に有効化できます。

Origin Shield は、CloudFront コンソール、AWS CloudFormation、または CloudFront API で有効にすることができます。

Console

既存のオリジンに対して Origin Shield を有効にするには (コンソール)

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 更新するオリジンがあるディストリビューションを選択します。
3. [Origin and Origin Groups (オリジンおよびオリジングループ)] タブを選択します。
4. 更新するオリジンを選択し、[編集] を選択します。
5. [Origin Shield を有効にする] で、[はい] を選択します。
6. [Origin Shield Region] (Origin Shield のリージョン) には、Origin Shield を有効にする AWS リージョンを選択します。リージョンの選択については、「Origin Shield の AWS リージョンの選択」を参照してください。
7. ページの最下部で [はい、編集します] を選択します。

ディストリビューションのステータスが [デプロイ済み] であれば、Origin Shield の準備が完了しています。これには数分かかります。

新しいオリジンの Origin Shield を有効にするには (コンソール)

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 既存のディストリビューションに新しいオリジンを作成するには、次の操作を行います。

   1. オリジンを作成するディストリビューションを選択します。
   2. [オリジンの作成] を選択し、ステップ 3 に進みます。

   新しいディストリビューションに新しいオリジンを作成するには、次の操作を行います。

   1. [Create Distribution] を選択します。
   2. [ウェブ] セクションで、[今すぐ始める] を選択します。[オリジン設定] セクションで、次の操作をステップ 3 から行います。
3. [Origin Shield を有効にする] で、[はい] を選択します。
4. [Origin Shield Region] (Origin Shield のリージョン) には、Origin Shield を有効にする AWS リージョンを選択します。リージョンの選択については、「Origin Shield の AWS リージョンの選択」を参照してください。

   新しいディストリビューションを作成する場合は、そのページの他の設定を使用して、ディストリビューションの設定を続けて行います。詳細については、「ディストリビューション設定リファレンス」を参照してください。
5. [作成] (既存のディストリビューションに新しいオリジンを作成する場合) を選択するか、[ディストリビューションの作成] (新しいディストリビューションに新しいオリジンを作成する場合) を選択して、変更内容を保存します。

ディストリビューションのステータスが [デプロイ済み] であれば、Origin Shield の準備が完了しています。これには数分かかります。

AWS CloudFormation

AWS CloudFormation で Origin Shield を有効にするには、`AWS::CloudFront::Distribution` リソースの `Origin` プロパティタイプに `OriginShield` プロパティを使用します。`OriginShield` プロパティは、既存の `Origin` に追加することも、新しい `Origin` を作成するときに含めることができます。

次の例は、米国西部 (オレゴン) リージョン (OriginShield) で us-west-2 を有効にするための構文を YAML 形式で示しています。リージョンの選択については、「the section called “Origin Shield の AWS リージョンの選択”」を参照してください。この例では、Origin リソース全体ではなく、AWS::CloudFront::Distribution プロパティタイプのみを示しています。

```
Origins:
- DomainName: 3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com
  Id: Example-EMP-3ae97e9482b0d011
  OriginShield:
    Enabled: true
    OriginShieldRegion: us-west-2
  CustomOriginConfig:
    OriginProtocolPolicy: match-viewer
    OriginSSLProtocols: TLSv1
```

詳細については、AWS CloudFormation ユーザーガイドのリソースとプロパティのリファレンスセクションで「AWS::CloudFront::Distribution Origin」を参照してください。

API

AWS SDK または AWS Command Line Interface (AWS CLI) を使用して CloudFront API で Origin Shield を有効にするには、OriginShield タイプを使用します。OriginShield は、Origin の DistributionConfig 内で指定します。OriginShield タイプの詳細については、「Amazon CloudFront API リファレンス」の以下の情報を参照してください。

- OriginShield (タイプ)
- Origin (タイプ)
- DistributionConfig (タイプ)
- UpdateDistribution (オペレーション)
- CreateDistribution (オペレーション)

これらのタイプと操作を使用するための具体的な構文は、使用する SDK、CLI、API クライアントによって異なります。詳細については、SDK、CLI、またはクライアントのリファレンスドキュメントを参照してください。

Origin Shield のコストの見積もり

Origin Shield の料金は、増分レイヤーとして Origin Shield に送信されるリクエストの数に基づいて計算されます。

オリジンにプロキシ化される動的 (キャッシュ不可能な) リクエストの場合、Origin Shield は常に増分レイヤーとなります。動的リクエストでは、HTTP メソッド PUT、POST、PATCH、DELETE を使用します。

有効期限 (TTL) の設定が 3,600 秒未満の GET および HEAD リクエストは、動的リクエストと見なされます。また、キャッシュが無効になっている GET および HEAD リクエストも動的リクエストと見なされます。

動的リクエストに対する Origin Shield の料金を見積もるには、次の式を使用します。

動的リクエストの総数 x 10,000 リクエストあたりの Origin Shield 料金 / 10,000

HTTP メソッド GET、HEAD、OPTIONS を使用する非動的リクエストの場合、Origin Shield は増分レイヤーになることがあります。Origin Shield を有効にするときは、Origin Shield の AWS リージョンを選択します。Origin Shield と同じリージョン内のリージョン別エッジキャッシュに本来送信されるリクエストの場合、Origin Shield は増分レイヤーになりません。このようなリクエストに対しては、Origin Shield の料金が発生しません。Origin Shield とは異なるリージョンのリージョン別エッジキャッシュに送信されてから Origin Shield に送信されるリクエストの場合、Origin Shield は増分レイヤーになります。このようなリクエストに対しては、Origin Shield の料金が発生します。

キャッシュ可能なリクエストに対する Origin Shield の料金を見積もるには、次の式を使用します。

キャッシュ可能なリクエストの総数 x (1 – キャッシュヒット率) x 異なるリージョンのリージョン別エッジキャッシュから Origin Shield に送信されるリクエストの割合 x 10,000 リクエストあたりの Origin Shield 料金 / 10,000

Origin Shield の 10,000 リクエストあたりの料金について詳しくは、「CloudFront の料金」を参照してください。

Origin Shield の高可用性

Origin Shield では、CloudFront のリージョン別エッジキャッシュを活用します。これらのエッジキャッシュはそれぞれ、少なくとも 3 つのアベイラビリティーゾーンと Amazon EC2 Auto Scaling インスタンスのフリートを使用して、AWS リージョン内に構築されます。CloudFront のロケーションから Origin Shield への接続では、リクエストごとにアクティブなエラー追跡が使用されます。こ

れにより、プライマリ Origin Shield ロケーションが利用できない場合は、リクエストがセカンダリ Origin Shield ロケーションに自動的にルーティングされます。

Origin Shield と他の CloudFront 機能との連携

以下のセクションでは、Origin Shield と他の CloudFront 機能との連携について説明します。

Origin Shield と CloudFront ログ記録

Origin Shield がいつリクエストを処理したかを確認するには、以下のいずれかを有効にする必要があります。

- CloudFront 標準ログ (アクセスログ)。標準ログは無料で提供されます。
- CloudFront リアルタイムログ。リアルタイムログの使用には追加料金が発生します。「Amazon CloudFront の料金」を参照してください。

Origin Shield からのキャッシュヒットは、CloudFront ログの `OriginShieldHit` フィールドに `x-edge-detailed-result-type` として表示されます。Origin Shield では、Amazon CloudFront のリージョン別エッジキャッシュが活用されます。リクエストが CloudFront エッジロケーションから Origin Shield として機能するリージョン別エッジキャッシュにルーティングされた場合、ログには `Hit` としてではなく `OriginShieldHit` として報告されます。

Origin Shield とオリジングループ

Origin Shield は CloudFront オリジングループとの互換性があります。Origin Shield はオリジンのプロパティであるため、オリジンがオリジングループの一部であっても、リクエストは常に各オリジンの Origin Shield を通過します。CloudFront はリクエストごとに、プライマリオリジンの Origin Shield を介してリクエストをオリジングループのプライマリオリジンにルーティングします。このリクエストが失敗した場合 (オリジングループのフェイルオーバー基準に従って)、CloudFront はセカンダリオリジンの Origin Shield を介してリクエストをセカンダリオリジンにルーティングします。

Origin Shield と Lambda@Edge

Origin Shield は Lambda@Edge 関数の機能性には影響しませんが、これらの関数が実行される AWS リージョンに影響を及ぼす可能性があります。

Lambda@Edge と共に Origin Shield を使用する場合、オリジン向けのトリガー (オリジンリクエストとオリジンレスポンス) は、Origin Shield が有効になっている AWS リージョンで実行されます。プライマリ Origin Shield ロケーションが利用できないために、CloudFront がリクエストをセカンダ

リ Origin Shield ロケーションにルーティングする場合、Lambda@Edge オリジン向けのトリガーもセカンダリ Origin Shield ロケーションを使用するようにシフトされます。

ビューワー向けトリガーは影響を受けません。

CloudFront オリジンフェイルオーバーを使用して高可用性を最適化する

高可用性が必要なシナリオでは、オリジンフェイルオーバーを使用するように CloudFront を設定できます。開始するには、プライマリとセカンダリの 2 つのオリジンを持つオリジングループを作成します。プライマリオリジンが使用できない場合、または障害を示す特定の HTTP レスポンスステータスコードを返す場合、CloudFront は自動的にセカンダリオリジンに切り替わります。

オリジンフェイルオーバーを設定するには、少なくとも 2 つのオリジンを持つディストリビューションが必要です。次に、1 つをプライマリとして設定した 2 つのオリジンを含むディストリビューションのオリジングループを作成します。最後に、オリジングループを使用するようにキャッシュ動作を作成または更新します。

オリジングループを設定して特定のオリジンフェイルオーバーオプションを設定する手順については、「オリジングループを作成する」を参照してください。

キャッシュ動作のオリジンフェイルオーバーを設定すると、CloudFront は、ビューワーリクエストに対して以下の処理を実行します。

- キャッシュヒットがあると、CloudFront はリクエストされたオブジェクトを返します。
- キャッシュミスがあると、CloudFront はオリジングループのプライマリオリジンにリクエストをルーティングします。
- プライマリオリジンがフェイルオーバー用に設定されていないステータスコード (HTTP 2xx や 3xx ステータスコードなど) を返すと、CloudFront はリクエストされたオブジェクトをビューワーに配信します。
- 次のいずれかに該当する場合:
  - プライマリオリジンは、フェイルオーバー用に設定した HTTP ステータスコードを返す
  - CloudFront がプライマリオリジンへの接続に失敗する
  - プライマリオリジンからの応答に時間がかかりすぎる (タイムアウト)

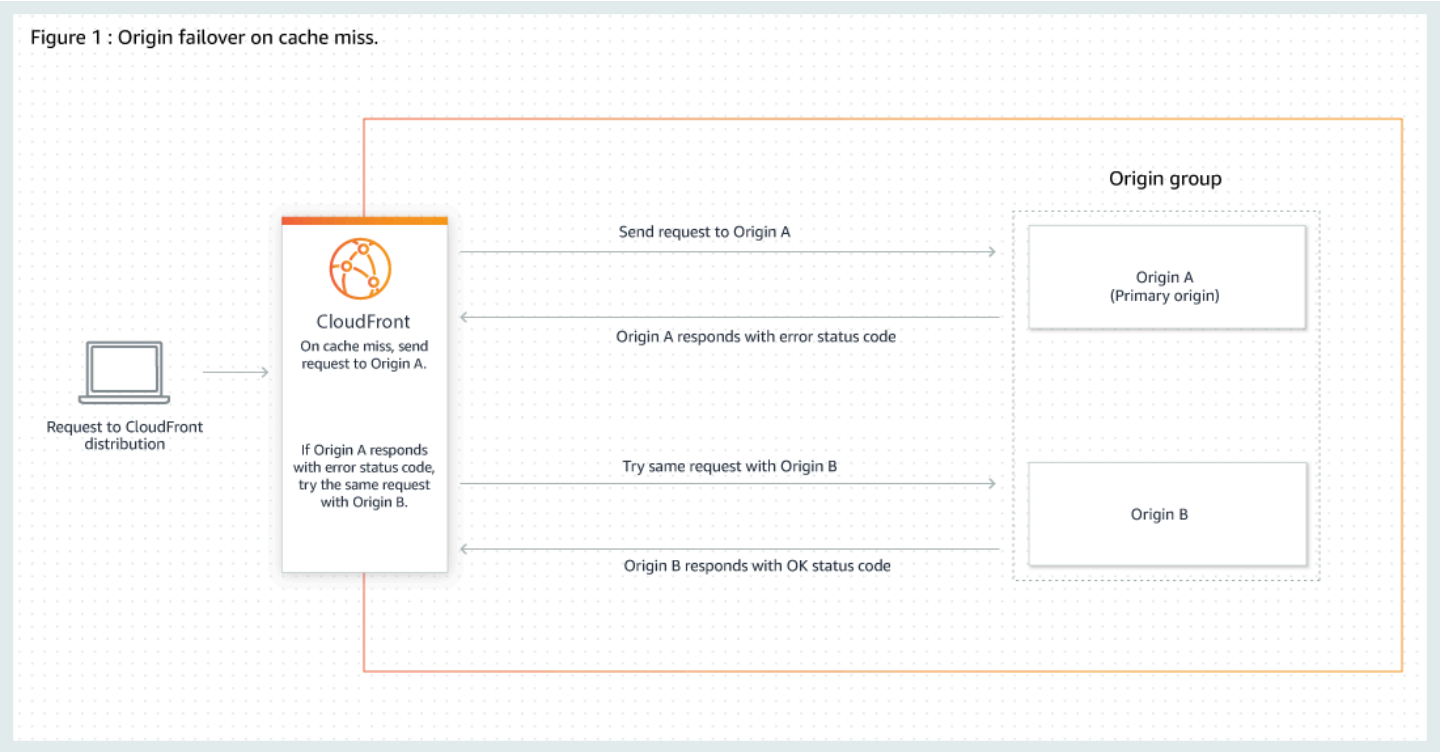
  この場合、CloudFront はオリジングループ内のセカンダリオリジンにリクエストをルーティングします。

> **Note**
>
> ビデオコンテンツのストリーミングなど、ユースケースによっては、CloudFront をセカンダリオリジンにすばやくフェイルオーバーすることが必要になる場合があります。CloudFront がセカンダリオリジンにフェイルオーバーするのにかかる時間を調整するには、「オリジンのタイムアウトと試行を制御する」を参照してください。

CloudFront は、前のリクエストをセカンダリオリジンにフェイルオーバーした場合でも、すべての受信リクエストをプライマリオリジンにルーティングします。CloudFront は、プライマリオリジンへのリクエストが失敗した後にのみセカンダリオリジンにリクエストを送信します。

CloudFront は、ビューワーリクエストの HTTP メソッドが GET、HEAD、または OPTIONS の場合にのみ、セカンダリオリジンにフェイルオーバーします。ビューワーが別の HTTP メソッド (たとえば POST や PUT など) を送信しても、CloudFront はフェイルオーバーしません。

以下の図は、オリジンフェイルオーバーのしくみを示しています。



トピック

- オリジングループを作成する

- オリジンのタイムアウトと試行を制御する
- Lambda@Edge 関数でのオリジンフェイルオーバーの使用
- オリジンフェイルオーバーでのカスタムエラーページの使用

オリジングループを作成する

オリジングループを作成するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. オリジングループを作成するディストリビューションを選択します。
3. [オリジン] タブを選択します。
4. ディストリビューションに複数のオリジンがあることを確認します。そうでない場合は、2 番目のオリジンを追加します。
5. [Origin groups] (オリジンのグループ) ペインの [Origins] (オリジン) タブで、[Create Origin group] (オリジングループの作成) を選択します。
6. オリジングループのオリジンを選択します。オリジンを追加したら、矢印を使用して優先度 (つまり、どのオリジンがプライマリで、どのオリジンがセカンダリであるか) を設定します。
7. オリジングループの名前を入力します。
8. フェイルオーバー基準として使用する HTTP ステータスコードを選択します。次のステータスコードを任意に組み合わせて選択できます。400、403、404、416、500、502、503、または 504。CloudFront は、指定したステータスコードの 1 つを含むレスポンスを受信すると、セカンダリオリジンにフェイルオーバーします。

   > **Note**
   >
   > CloudFront は、ビューワーリクエストの HTTP メソッドが GET、HEAD、または OPTIONS の場合にのみ、セカンダリオリジンにフェイルオーバーします。ビューワーが別の HTTP メソッド (たとえば POST や PUT など) を送信しても、CloudFront はフェイルオーバーしません。

9. [Create origin group] (オリジングループの作成) を選択します。

ディストリビューションのキャッシュビヘイビアのオリジンとしてオリジングループを割り当ててください。詳細については、「名前」を参照してください。

オリジンのタイムアウトと試行を制御する

デフォルトでは、CloudFront は、セカンダリオリジンにフェイルオーバーする前に 30 秒間 (それぞれ 10 秒間の接続試行が 3 回)、オリジングループ内のプライマリオリジンへの接続を試行します。ビデオコンテンツのストリーミングなど、ユースケースによっては、CloudFront をセカンダリオリジンに、さらにすばやくフェイルオーバーすることが必要になる場合があります。以下の設定を調整して、CloudFront がセカンダリオリジンにフェイルオーバーするのにかかる時間を変更できます。オリジンがセカンダリオリジンであるか、オリジングループの一部ではないオリジンである場合、これらの設定は、CloudFront が HTTP 504 レスポンスをビューワーに返す速度に影響します。

よりすばやくフェイルオーバーするには、接続タイムアウトを短くするか、接続試行回数を減らすか、またはその両方を行います。カスタムオリジン (静的ウェブサイトホスティングで設定されている Amazon S3 バケットオリジンを含む) の場合、オリジン応答タイムアウトを調整することもできます。

オリジン接続タイムアウト

オリジン接続タイムアウト設定は、オリジンへの接続を確立しようとしたときに CloudFront が待機する時間に影響します。デフォルトでは、CloudFront は接続の確立まで 10 秒待機しますが、1 ~ 10 秒 (両端の値を含む) を指定できます。詳細については、「接続タイムアウト」を参照してください。

オリジン接続の試行

オリジン接続試行の設定は、CloudFront がオリジンへの接続を試行する回数に影響します。デフォルトでは、CloudFront は 3 回接続を試行しますが、1 ~ 3 (両端の値を含む) を指定できます。詳細については、「接続の試行」を参照してください。

カスタムオリジン (静的ウェブサイトホスティングで設定された Amazon S3 バケットを含む) では、この設定は、オリジン応答タイムアウトの場合に CloudFront がオリジンから応答を取得しようとする回数にも影響します。

オリジン応答タイムアウト

> **Note**
>
> これは、カスタムオリジンにのみ適用されます。

オリジン応答タイムアウト設定は、CloudFront がオリジンからのレスポンスを受信する (または完全なレスポンスを受信する) のを待機する時間の長さに影響します。デフォルトで

は、CloudFront は 30 秒間待機しますが、1 ~ 60 秒 (両端の値を含む) を指定できます。詳細については、「応答タイムアウト (カスタムオリジンのみ)」を参照してください。

これらの設定を変更する方法

CloudFront コンソールでこれらの設定を変更するには

- 新しいオリジンまたは新しいディストリビューションの場合、リソースの作成時にこれらの値を指定します。
- 既存のディストリビューションの既存のオリジンについては、オリジンを編集するときにこれらの値を指定します。

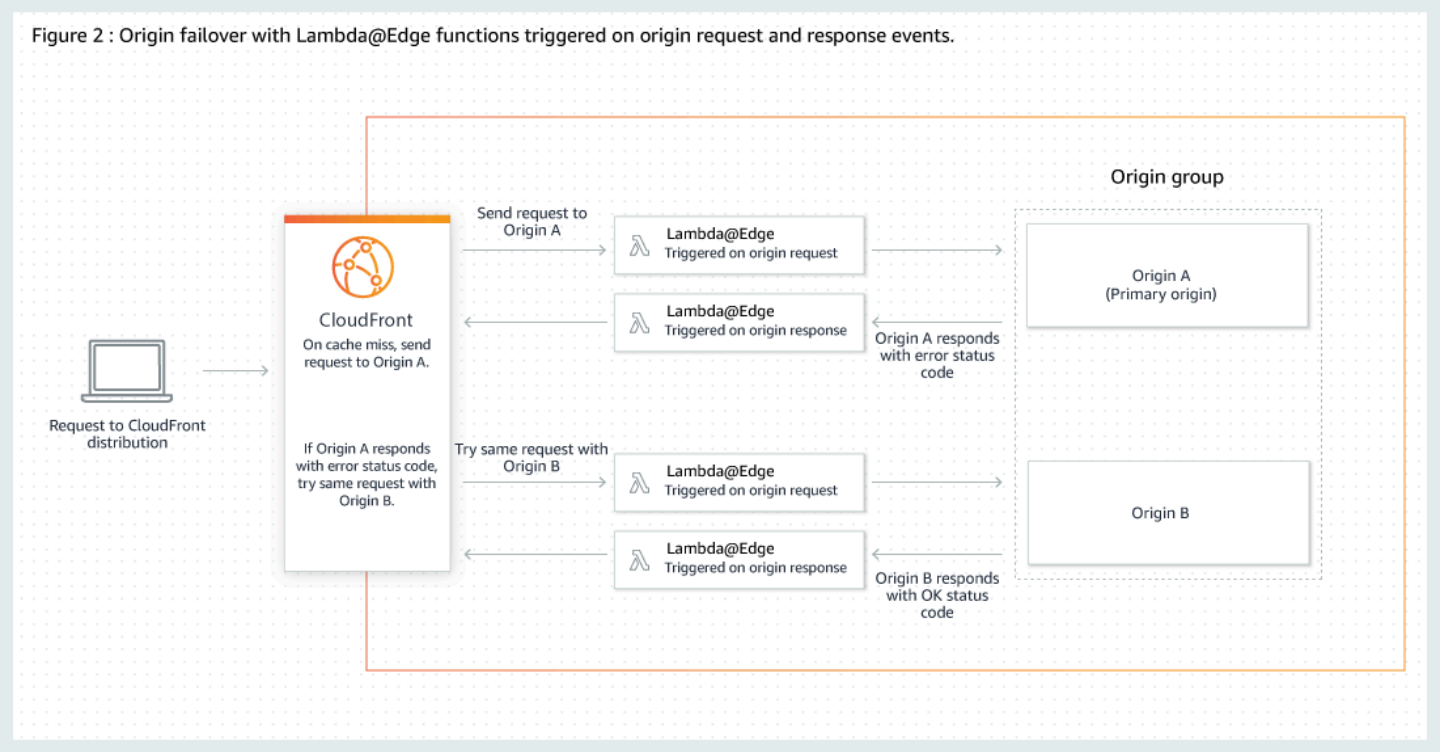
詳細については、「ディストリビューション設定リファレンス」を参照してください。

Lambda@Edge 関数でのオリジンフェイルオーバーの使用

Lambda@Edge 関数は、オリジングループで設定した CloudFront ディストリビューションで使用できます。Lambda 関数を使用するには、キャッシュ動作を作成するときにオリジングループのオリジンリクエストまたはオリジンレスポンストリガーで指定します。オリジングループで Lambda@Edge 関数を使用すると、1 つのビューワーリクエストに対して、この関数を 2 回トリガーできます。たとえば、次のシナリオが考えられます。

1. オリジンリクエストトリガーを使用して Lambda@Edge 関数を作成します。
2. Lambda 関数は、(キャッシュミス時に) プライマリオリジンに CloudFront がリクエストを送信したときに 1 回トリガーされます。
3. プライマリオリジンは、フェイルオーバー用に設定された HTTP ステータスコードで応答します。
4. CloudFront がセカンダリオリジンに同じリクエストを送信すると、Lambda 関数が再度トリガーされます。

次の図は、オリジンリクエストまたはレスポンストリガーに Lambda@Edge 関数を含める場合、オリジンフェイルオーバーがどのように機能するかを示しています。



Lambda@Edge トリガーの使用の詳細については、「the section called “Lambda@Edge 関数のトリガーを追加する”」を参照してください。

DNS フェイルオーバーの管理の詳細については、「Amazon Route 53 デベロッパーガイド」の「DNS フェイルオーバーの設定」を参照してください。

オリジンフェイルオーバーでのカスタムエラーページの使用

オリジンフェイルオーバー用に設定されていないオリジンでそれらを使用する方法と同様に、オリジングループでカスタムエラーページを使用できます。

オリジンフェイルオーバーを使用すると、プライマリまたはセカンダリオリジン (またはその両方) のカスタムエラーページを返すように CloudFront を設定できます。

- プライマリオリジンのカスタムエラーページを返す - プライマリオリジンが、フェイルオーバー用に設定されていない HTTP ステータスコードを返す場合、CloudFront はカスタムエラーページをビューワーに返します。
- セカンダリオリジンのカスタムエラーページを返す - CloudFront がセカンダリオリジンからエラーステータスコードを受け取った場合、CloudFront はカスタムエラーページを返します。

CloudFront でカスタムエラーページを使用する方法の詳細については、「カスタムエラーレスポンスを生成する」を参照してください。

コンテンツをキャッシュに保持する期間 (有効期限) を管理する

CloudFront が別のリクエストをオリジンに転送するまでにファイルを CloudFront キャッシュに保持する期間を制御できます。この期間を短くすると、動的なコンテンツを供給できます。この期間を長くすると、ユーザー側のパフォーマンスは向上します。ファイルがエッジキャッシュから直接返される可能性が高くなるためです。期間を長くすると、オリジンの負荷も軽減されます。

一般的に CloudFront は、指定したキャッシュ保持期間が経過するまで、つまりファイルの有効期限が切れるまでエッジロケーションからファイルを提供します。ファイルの有効期限が切れると、エッジロケーションがファイルのリクエストを次に受け取ったときに、CloudFront は、リクエストをオリジンに転送し、キャッシュにファイルの最新バージョンが含まれていることを確認します。オリジンからのレスポンスは、ファイルが変更されたかどうかによって異なります。

- CloudFront キャッシュに最新バージョンがすでにある場合、オリジンはステータスコード `304 Not Modified` を返します。
- CloudFront キャッシュに最新バージョンがない場合、オリジンはステータスコード `200 OK` とファイルの最新バージョンを返します。

エッジロケーションに頻繁にリクエストされないファイルがあれば、CloudFront は、頻繁にリクエストされるようになったファイル用にスペースを確保するために、そのファイルを削除する (そのファイルの有効期限が切れる前に削除する) 場合があります。

デフォルトでは、各ファイルは 24 時間後に自動的に有効期限切れになりますが、2 つの方法でこのデフォルトの動作を変更できます。

- 同じパスパターンに一致するすべてのファイルのキャッシュ保持期間を変更するには、CloudFront の設定でキャッシュの動作の [Minimum TTL (最小 TTL)]、[Maximum TTL (最大 TTL)]、[Default TTL (デフォルト TTL)] を変更できます。個々の設定については、「the section called “ディストリビューションの設定”」の「最小 TTL」、「最大 TTL」、「デフォルト TTL」を参照してください。
- 個々のファイルのキャッシュ保持期間を変更するには、ファイルに `Cache-Control` または `max-age` ディレクティブが付いた `s-maxage` を追加するか、`Expires` ヘッダーフィールドを追加するようにオリジンを設定します。詳細については、「ヘッダーを使用して個々のオブジェクトのキャッシュ保持期間を制御する」を参照してください。

[Minimum TTL (最小 TTL)]、[Default TTL (デフォルト TTL)]、[Maximum TTL (最大 TTL)] が max-age ディレクティブ、s-maxage ディレクティブ、Expires ヘッダーフィールドとどのように連動するかの詳細については、「the section called “CloudFront がオブジェクトをキャッシュする期間を指定する”」を参照してください。

CloudFront がオリジンに別のリクエストを転送して、リクエストされたオブジェクトを再度取得することを試みるまでに、エラー (404 Not Found など) が CloudFront キャッシュに保持される期間を制御することもできます。詳細については、「the section called “CloudFront がオリジンからの HTTP 4xx および 5xx ステータスコードを処理する方法”」を参照してください。

トピック

- ヘッダーを使用して個々のオブジェクトのキャッシュ保持期間を制御する
- 古い (期限切れの) コンテンツを提供する
- CloudFront がオブジェクトをキャッシュする期間を指定する
- Amazon S3 コンソールを使用してオブジェクトにヘッダーを追加する

ヘッダーを使用して個々のオブジェクトのキャッシュ保持期間を制御する

Cache-Control および Expires ヘッダーを使用して、オブジェクトをキャッシュに保持する期間を制御できます。[Minimum TTL]、[Default TTL]、[Maximum TTL] の設定もキャッシュ保持期間に影響を与えますが、ここでは、ヘッダーがキャッシュ保持期間に与える影響について概要を示します。

- Cache-Control max-age ディレクティブでは、CloudFront がオリジンサーバーからオブジェクトを再度取得するまでにオブジェクトをキャッシュに保持する期間 (秒単位) を指定できます。CloudFront がサポートする最小有効期限は 0 秒です。最大値は 100 (年) です。値は次の形式で指定します。

  ```
  Cache-Control: max-age=#
  ```

  例えば、以下のディレクティブは CloudFront に関連付けられているオブジェクトを 3,600 秒 (1 時間) キャッシュに保持するよう指示します。

  ```
  Cache-Control: max-age=3600
  ```

  ブラウザキャッシュに保持される期間とは異なる期間、オブジェクトを CloudFront エッジキャッシュに保持する場合、Cache-Control max-age ディレクティブと Cache-Control s-maxage ディレクティブを併用できます。詳細については、「CloudFront がオブジェクトをキャッシュする期間を指定する」を参照してください。

- Expires ヘッダーフィールドでは、「RFC 2616、ハイパーテキスト転送プロトコル — HTTP/1.1 セクション 3.3.1、完全な日付」に規定された形式を使用して、有効期限切れ日時を指定できます。

  ```
  Sat, 27 Jun 2015 23:59:59 GMT
  ```

オブジェクトのキャッシュを制御するには、Cache-Control max-age ヘッダーフィールドではなく、Expires ディレクティブを使用することをお勧めします。Cache-Control max-age と Expires の両方の値を指定した場合、CloudFront は Cache-Control max-age の値のみを使用します。

詳細については、「CloudFront がオブジェクトをキャッシュする期間を指定する」を参照してください。

ビューワーからの Cache-Control リクエストで HTTP Pragma または GET ヘッダーフィールドを使用して、オリジンサーバーに戻ってオブジェクトを取得するように CloudFront を設定することはできません。CloudFront は、ビューワーからのリクエストにあるそのようなヘッダーフィールドを無視します。

Cache-Control および Expires ヘッダーフィールドの詳細については、「RFC 2616、ハイパーテキスト転送プロトコル — HTTP/1.1」の以下のセクションを参照してください。

- セクション 14.9 キャッシュ制御
- セクション 14.21 期限

古い (期限切れの) コンテンツを提供する

CloudFront は Stale-While-Revalidate および Stale-If-Error キャッシュ制御ディレクティブをサポートしています。

- この stale-while-revalidate ディレクティブにより、CloudFront はオリジンから新しいバージョンを非同期で取得しながら、キャッシュから古いコンテンツを提供できます。これにより、ユーザーはバックグラウンドでの取得を待たずに CloudFront のエッジロケーションからすぐにレスポンスを受け取り、今後のリクエストに備えて新しいコンテンツがバックグラウンドで読み込まれるため、レイテンシーが向上します。

  次の例では、CloudFront はレスポンスを 1 時間 (max-age=3600) キャッシュします。この期間を過ぎてリクエストが行われた場合、CloudFront は古いコンテンツを提供すると同時に、キャッ

シュされたコンテンツを再検証して更新するリクエストをオリジンに送信します。コンテンツが再検証される間、古いコンテンツは最大 10 分間 (stale-while-revalidate=600) 提供されます。

```
Cache-Control: max-age=3600, stale-while-revalidate=600
```

- オリジンにアクセスできない場合やエラーコード 500～600 が返された場合、stale-if-error ディレクティブにより、CloudFront はキャッシュから古いコンテンツを提供できます。これにより、ビューワーはオリジンが停止しているときでもコンテンツにアクセスできるようになります。

  次の例では、CloudFront はレスポンスを 1 時間 (max-age=3600) キャッシュします。この期間を過ぎてもオリジンがダウンしたり、エラーが返されたりした場合、CloudFront は最長 24 時間 (stale-if-error=86400)、古いコンテンツを提供し続けます。

  ```
  Cache-Control: max-age=3600, stale-if-error=86400
  ```

  > **Note**
  >
  > stale-if-error およびカスタムエラーレスポンスの両方が設定されると、指定された stale-if-error 期間内にエラーが発生した場合、CloudFront はまず古いコンテンツの提供を試みます。古いコンテンツが利用できない場合、またはコンテンツが stale-if-error の期間を超えている場合、CloudFront は対応するエラーステータスコードに設定されたカスタムエラーレスポンスを提供します。

両方を一緒に使用する

stale-while-revalidate および stale-if-error は独立したキャッシュ制御ディレクティブで、これらを一緒に使用することでレイテンシーを減らしたり、オリジンが応答または回復するためのバッファを追加したりできます。

次の例では、CloudFront はレスポンスを 1 時間 (max-age=3600) キャッシュします。この期間を過ぎてからリクエストが行われた場合、コンテンツが再検証される間、CloudFront は古いコンテンツを最大 10 分間 (stale-while-revalidate=600) 提供します。CloudFront がコンテンツを再検証しようとしたときにオリジンサーバーがエラーを返した場合、CloudFront は古いコンテンツを最大 24 時間 (stale-if-error=86400) 提供し続けます。

```
Cache-Control: max-age=3600, stale-while-revalidate=600, stale-if-error=86400
```

> **Tip**
>
> キャッシュによって、パフォーマンスと鮮度が保たれます。stale-while-revalidate や stale-if-error などのディレクティブを使用すると、パフォーマンスとユーザーエクスペリエンスが向上しますが、コンテンツをどれだけ新鮮にするかの希望に合った設定にしてください。古いコンテンツディレクティブは、コンテンツを更新する必要があるが、最新バージョンであることが重要でない場合に最適です。さらに、コンテンツが変更されないか、ほとんど変更されない場合、stale-while-revalidate は不要なネットワークリクエストを追加する可能性があります。代わりに、キャッシュ期間を長く設定することを検討してください。

CloudFront がオブジェクトをキャッシュする期間を指定する

CloudFront が、オリジンに別のリクエストを送信するまでの期間に オブジェクトをキャッシュに保持する時間の長さを制御するには、次の方法があります。

- CloudFront ディストリビューションのキャッシュ動作の TTL 値に、最小、最大、およびデフォルトの値を設定します。これらの値は、キャッシュ動作にアタッチされたキャッシュポリシー (推奨) の中、またはレガシーキャッシュ設定の中で設定できます。
- オリジンからの応答に Cache-Control または Expires ヘッダーを含めます。これらのヘッダーは、別のリクエストがブラウザから CloudFront に送信されるまでの期間に、オブジェクトがブラウザーキャッシュに保持される時間を定義するためにも役立ちます。

次の表では、オリジンから送信された Cache-Control ヘッダーと Expires ヘッダーがキャッシュ動作の TTL 設定とどのように関係し、キャッシュに影響を与えるのかを説明しています。

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
オリジンが **Cache-Control: max-age** ディレクティブをオブジェクトに追加	CloudFront キャッシュ CloudFront は、Cache-Control: max-age ディレクティブの値と最大 TTL の値のうち、小さい方の値に対応	CloudFront キャッシュ CloudFront のキャッシュ動作は、CloudFront 最小 TTL および最大 TTL、Cache-Control max-age ディレク

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
	する期間、オブジェクトをキャッシュに保持します。 ブラウザキャッシュ ブラウザは、Cache-Control: max-age ディレクティブの値に対応する期間、オブジェクトをキャッシュに保持します。	ティブの値によって異なります。 • 最小 TTL < max-age < 最大 TTL である場合、CloudFront は Cache-Control: max-age ディレクティブの値に対応する期間、オブジェクトをキャッシュに保持します。 • max-age < 最小TTL の場合、CloudFront は最小 TTL の値に対応する期間、オブジェクトをキャッシュに保持します。 • max-age > 最大 TTL である場合、CloudFront は最大 TTL の値に対応する期間、オブジェクトをキャッシュに保持します。 ブラウザキャッシュ ブラウザは、Cache-Control: max-age ディレクティブの値に対応する期間、オブジェクトをキャッシュに保持します。

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
オリジンが **Cache-Control: max-age** ディレクティブをオブジェクトに追加しない	CloudFront キャッシュ CloudFront は、デフォルト TTL の値に対応する期間、オブジェクトをキャッシュに保持します。 ブラウザキャッシュ ブラウザによって異なります。	CloudFront キャッシュ CloudFront は、最小 TTL またはデフォルト TTL の値のうち、大きい方の値に対応する期間、オブジェクトをキャッシュに保持します。 ブラウザキャッシュ ブラウザによって異なります。

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
オリジンが **Cache-Control: max-age** および **Cache-Control: s-maxage** ディレクティブをオブジェクトに追加	CloudFront キャッシュ CloudFront は、Cache-Control: s-maxage ディレクティブの値と最大 TTL の値のうち、小さい方の値に対応する期間、オブジェクトをキャッシュに保持します。 ブラウザキャッシュ ブラウザは、Cache-Control max-age ディレクティブの値に対応する期間、オブジェクトをキャッシュに保持します。	CloudFront キャッシュ CloudFront のキャッシュ動作は、CloudFront 最小 TTL および最大 TTL、Cache-Control: s-maxage ディレクティブの値によって異なります。 • 最小 TTL < s-maxage < 最大 TTL である場合、CloudFront は Cache-Control: s-maxage ディレクティブの値に対応する期間、オブジェクトをキャッシュに保持します。 • s-maxage < 最小TTL の場合、CloudFront は最小 TTL の値に対応する期間、オブジェクトをキャッシュに保持します。 • s-maxage > 最大 TTL である場合、CloudFront は最大 TTL の値に対応する期間、オブジェクトをキャッシュに保持します。 ブラウザキャッシュ ブラウザは、Cache-Control: max-age ディレク

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
		ティブの値に対応する期間、オブジェクトをキャッシュに保持します。

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
オリジンが **Expires** ヘッダーをオブジェクトに追加	CloudFront キャッシュ CloudFront は、Expires ヘッダーにある日付と最大 TTL の値に対応する日付のうち早い方の日付まで、オブジェクトをキャッシュに保持します。 ブラウザキャッシュ ブラウザは、Expires ヘッダーにある日付まで、オブジェクトをキャッシュに保持します。	CloudFront キャッシュ CloudFront キャッシュは、CloudFront 最小 TTL および最大 TTL、Expires ヘッダーの値によって異なります。 • 最小 TTL Expires < 最大 TTL である場合、CloudFront は Expires ヘッダーの日付けおよび時刻まで、オブジェクトをキャッシュに保持します。 • Expires < 最小TTL の場合、CloudFront は最小 TTL の値に対応する期間、オブジェクトをキャッシュに保持します。 • Expires > 最大 TTL である場合、CloudFront は最大 TTL の値に対応する期間、オブジェクトをキャッシュに保持します。 ブラウザキャッシュ ブラウザは、Expires ヘッダーの日付けおよび時刻まで、オブジェクトをキャッシュに保持します。

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
オリジンが **Cache-Control: no-cache** 、**no-store**、および (または) **private** ディレクティブをオブジェクトに追加	CloudFront とブラウザはヘッダーを優先させます。	CloudFront キャッシュ CloudFront は、最小 TTL の値に対応する期間、オブジェクトをキャッシュに保持します。この表の下にある注意点をお読みください。 ブラウザキャッシュ ブラウザはヘッダーを優先します。

> **Warning**
>
> 最小 TTL が 0 より大きい場合は、`Cache-Control: no-cache`、`no-store`、`private` ディレクティブがオリジンヘッダーに含まれていても、CloudFront はキャッシュポリシーの最小 TTL を使用します。
> オリジンとの接続が可能な場合、CloudFront はオリジンからオブジェクトを取得し、ビューワーに返します。
> オリジンが接続不能で、最小または最大 TTL 値が 0 より大きい場合、CloudFront は、以前にオリジンから取得したオブジェクトを返します。
> この動作を回避するには、`Cache-Control: stale-if-error=0` ディレクティブに、オリジンから返されたオブジェクトを含めます。このようにすることで、オリジンが接続不能な場合に CloudFront が以後のリクエストに応答する際、以前にオリジンから取得したオブジェクトを返すのではなくエラーを返すようになります。

CloudFront コンソールを使用して、ディストリビューションの設定を変更する方法については、「ディストリビューションを更新する」を参照してください。CloudFront API を使用してディストリビューションの設定を変更する方法については、「UpdateDistribution」を参照してください。

Amazon S3 コンソールを使用してオブジェクトにヘッダーを追加する

Amazon S3 コンソールを使用して Amazon S3 オブジェクトに **Cache-Control** または **Expires** ヘッダーフィールドを追加するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. バケットの一覧で、ヘッダーを追加するファイルを含むバケットの名前を選択します。
3. ヘッダーを追加するファイルまたはフォルダの名前の横にあるチェックボックスをオンにします。フォルダにヘッダーを追加すると、そのフォルダ内のすべてのファイルに影響します。
4. [Actions] (アクション) を選択し、[Edit metadata] (メタデータの編集) を選択します。
5. [Add metadata] (メタデータを追加) パネルで 、次の操作を行います。
   a. [Add metadata] (メタデータの追加) を選択します。
   b. [Type] (タイプ) で、[System defined] (システム定義) を選択します。
   c. [Key] (キー) で、追加するヘッダーの名前 ([Cache-Control] または [Expires]) を選択します。
   d. [Value] (値) で、ヘッダー値を入力します。例えば、`Cache-Control` ヘッダーの場合は、`max-age=86400` と入力します。`Expires` で、有効期限の日時を `Wed, 30 Jun 2021 09:28:00 GMT` のように入力できます。
6. ページの最下部で [Edit metadata] (メタデータの編集) を選択します。

クエリ文字列パラメータに基づいてコンテンツをキャッシュする

ウェブアプリケーションによっては、クエリ文字列を使用してオリジンに情報を送信します。クエリ文字列はウェブリクエストの一部で、? 文字の後に追加されます。この文字列には & 文字で区切られたパラメータを 1 つ以上含めることができます。次の例では、クエリ文字列には 2 つのパラメータ (*`color=red`* と *`size=large`*) が含まれています。

`https://d111111abcdef8.cloudfront.net/images/image.jpg?`*`color=red&size=large`*

ディストリビューションでは、CloudFront がクエリ文字列をオリジンに転送するかどうかと、すべてのパラメータまたは一部のパラメータのどちらに基づいてコンテンツをキャッシュするかを選択できます。これが役立つ場合があるのはなぜですか。次の の例を考えます。

たとえば、ウェブサイトが 5 種類の言語で使用でき、ディレクトリ構造とファイル名はウェブサイトの 5 つのバージョンすべてで共通だとします。ユーザーがウェブサイトを表示する

と、CloudFront に転送されるリクエストには、ユーザーが選択した言語に基づく言語によるクエリ文字列が含められます。また、クエリ文字列をオリジンに転送し、言語パラメータに基づいてキャッシュするよう CloudFront を設定できます。選択された言語に対応する特定バージョンのページを返すようウェブサーバーを設定した場合、CloudFront は、それぞれの言語によるクエリ文字列パラメータに基づく各言語のバージョンを個別にキャッシュします。

この例では、ウェブサイトのメインページが `main.html` であり、以下の 5 つのリクエストが実行されると、CloudFront は、各言語のクエリ文字列パラメータをそれぞれの値として `main.html` を 5 回キャッシュします。

- `https://d111111abcdef8.cloudfront.net/main.html?language=de`
- `https://d111111abcdef8.cloudfront.net/main.html?language=en`
- `https://d111111abcdef8.cloudfront.net/main.html?language=es`
- `https://d111111abcdef8.cloudfront.net/main.html?language=fr`
- `https://d111111abcdef8.cloudfront.net/main.html?language=jp`

次の点に注意してください。

- 一部の HTTP サーバーはクエリ文字列パラメータを処理しません。このため、パラメータ値に基づくオブジェクトの別バージョンを返しません。これらのオリジンに対して、クエリ文字列パラメータをオリジンに転送するように CloudFront を設定している場合、オリジンがパラメータ値ごとに同じバージョンのオブジェクトを CloudFront に返したとしても、CloudFront は引き続きパラメータ値に基づくキャッシュを実行します。
- 上記の例で説明したようにクエリ文字列パラメータを言語で使用するには、クエリ文字列パラメータ間の区切り文字として & 文字を使用する必要があります。別の区切り文字を使用した場合、キャッシュは、キャッシュ条件として CloudFront で指定するパラメータと、それらのパラメータがクエリ文字列に記述される順序の影響を受けて、予期しない結果が生じる場合があります。

  以下の例では、`color` パラメータだけに基づいてキャッシュするよう、異なる区切り記号を使用して CloudFront を設定した場合にどのようになるかを示しています。

  - 以下のリクエストでは、CloudFront は `color` パラメータの値に基づいてコンテンツをキャッシュしますが、この値を *`red;size=large`* と解釈します。

    ```
    https://d111111abcdef8.cloudfront.net/images/
    image.jpg?color=red;size=large
    ```

- 以下のリクエストでは、CloudFront はコンテンツをキャッシュしますが、実行されるキャッシュは、クエリ文字列パラメータに基づくものではありません。これは、color パラメータに基づいてキャッシュするよう CloudFront が設定されているが、CloudFront は以下の文字列に size パラメータ (値は *large;color=red*) のみが含まれていると解釈するためです。

  ```
  https://d111111abcdef8.cloudfront.net/images/
  image.jpg?size=large;color=red
  ```

次のいずれかを実行するよう CloudFront を設定できます。

- クエリ文字列をオリジンにまったく転送しない。クエリ文字列を転送しない場合、CloudFront はクエリ文字列パラメータに基づくキャッシュを実行しません。
- クエリ文字列をオリジンに転送し、クエリ文字列内のすべてのパラメータに基づいてキャッシュする。
- クエリ文字列をオリジンに転送し、クエリ文字列内の指定したパラメータに基づいてキャッシュする。

詳細については、「the section called “キャッシュを最適化する”」を参照してください。

トピック

- クエリ文字列の転送とキャッシュのためのコンソールおよび API の設定
- キャッシュを最適化する
- クエリ文字列パラメータと CloudFront 標準ログ (アクセスログ)

クエリ文字列の転送とキャッシュのためのコンソールおよび API の設定

クエリ文字列の転送とキャッシュを CloudFront コンソールで設定するには、「the section called “ディストリビューションの設定”」の以下のセクションを参照してください。

- the section called “クエリ文字列の転送とキャッシュ”
- the section called “クエリ文字列の許可リスト”

CloudFront の API を使用してクエリ文字列の転送とキャッシュを設定するには、Amazon CloudFront API リファレンスの「DistributionConfig」および「DistributionConfigWithTags」を参照してください。

- QueryString
- QueryStringCacheKeys

キャッシュを最適化する

クエリ文字列パラメータに基づいてキャッシュするように CloudFront を設定する場合、以下の手順を実行して、CloudFront がオリジンに転送するリクエストの数を減らすことができます。CloudFront エッジロケーションがオブジェクトを提供する場合、オリジンサーバーの負荷が軽減され、レイテンシーが減少します。これは、オブジェクトがユーザーに近い場所から提供されるためです。

オリジンが返すオブジェクトのバージョンが変わるパラメータだけに基づいてキャッシュする

CloudFront は、ウェブアプリケーションが CloudFront に転送する各クエリ文字列パラメータに対して、すべてのパラメータ値についてオリジンにリクエストを転送し、すべてのパラメータ値について異なるバージョンのオブジェクトをキャッシュします。これは、オリジンがパラメータの値に関係なく常に同じオブジェクトを返す場合も当てはまります。パラメータが複数ある場合、リクエスト数とオブジェクトの数は乗算されます。

このため、オリジンが返すバージョンが変化するようなクエリ文字列パラメータだけに基づいてキャッシュするよう CloudFront を設定し、各パラメータに基づいてキャッシュするメリットを慎重に検討することをお勧めします。たとえば、ある通販ウェブサイトを運営していて、ジャケットの写真が色違いで 6 つあり、ジャケットのサイズは 10 種類だとします。また、ジャケットの写真は色違いだけが表示され、サイズ違いの分までは表示されていないものとします。この場合にキャッシュを最適化するには、サイズのパラメータではなく、色のパラメータだけに基づいてキャッシュするよう CloudFront を設定する必要があります。これにより、CloudFront がキャッシュからリクエストを処理できる可能性が高くなり、パフォーマンスが向上し、オリジンの負荷が低下します。

パラメータの順序を常に統一する

クエリ文字列では、パラメータの順序が重要になります。次の例は、パラメータの順序だけが異なる同じクエリ文字列です。この場合 CloudFront は、image.jpg に対する 2 つの異なるリクエストをオリジンに転送し、2 つの異なるバージョンのオブジェクトをキャッシュします。

- https://d111111abcdef8.cloudfront.net/images/image.jpg?*color=red&size=large*
- https://d111111abcdef8.cloudfront.net/images/image.jpg?*size=large&color=red*

このため、パラメータ名は、常に同じ順序 (アルファベット順など) にすることをお勧めします。

パラメータ名とパラメータ値の大文字と小文字を常に統一する

CloudFront は、クエリ文字列パラメータに基づいてキャッシュを実行する際に、パラメータ名とパラメータ値の大文字と小文字の違いを区別します。次の例は、パラメータ名とパラメータ値の大文字と小文字だけが異なる、同じクエリ文字列です。この場合 CloudFront は、image.jpg に対する 4 つの異なるリクエストをオリジンに転送し、4 つの異なるバージョンのオブジェクトをキャッシュします。

- `https://d111111abcdef8.cloudfront.net/images/image.jpg?color=red`
- `https://d111111abcdef8.cloudfront.net/images/image.jpg?color=Red`
- `https://d111111abcdef8.cloudfront.net/images/image.jpg?Color=red`
- `https://d111111abcdef8.cloudfront.net/images/image.jpg?Color=Red`

このため、パラメータ名とパラメータ値の大文字と小文字を統一する (すべて小文字など) ことをお勧めします。

署名付き URL と競合するパラメータ名を使わない

署名付き URL を使用してコンテンツへのアクセスを制限している場合 (信頼された署名者をディストリビューションに追加した場合)、CloudFront は以下のクエリ文字列パラメータを削除してから URL の残りをオリジンに転送します。

- `Expires`
- `Key-Pair-Id`
- `Policy`
- `Signature`

署名付き URL を使用しており、クエリ文字列をオリジンに転送するように CloudFront を設定する場合、独自のクエリ文字列パラメータに `Expires`、`Key-Pair-Id`、`Policy`、または `Signature` という名前を付けることはできません。

クエリ文字列パラメータと CloudFront 標準ログ (アクセスログ)

ログ作成を有効にした場合、CloudFront は、クエリ文字列パラメータを含む完全な URL を ログに記録します。クエリ文字列がオリジンに転送されるように CloudFront を設定したかどうかに関係なく、そのような動作になります。CloudFront ログ記録の詳細については、「the section called “標準ログ (アクセスログ) の使用”」を参照してください。

Cookie に基づいてコンテンツをキャッシュする

デフォルトでは、CloudFront はリクエストとレスポンスを処理するとき、またはオブジェクトをエッジロケーションにキャッシュするときに Cookie を考慮しません。CloudFront が `Cookie` ヘッダーに含まれる内容以外は同一の 2 つのリクエストを受信した場合、デフォルトでは、CloudFront はリクエストを同一として扱い、両方のリクエストに対して同じオブジェクトを返します。

CloudFront でビューワーリクエストの一部またはすべて Cookie をオリジンに転送し、転送される Cookie 値に基づいてオブジェクトの別バージョンをキャッシュするように設定できます。これを行うと、CloudFront は、ビューワーリクエスト内の Cookie の一部またはすべて (転送するように設定されているものすべて) を使用して、キャッシュ内のオブジェクトを一意に識別します

たとえば、`locations.html` に対するリクエストに `country` Cookie が含まれており、その値が `uk` または `fr` であるとします。`country` Cookie の値に基づいてオブジェクトをキャッシュするように CloudFront を設定すると、CloudFront は `locations.html` に関するリクエストをオリジンに転送し、`country` Cookie とその値を含めます。オリジンは `locations.html` を返し、CloudFront は `country` Cookie の値が `uk` であるリクエスト用に 1 回、値が `fr` であるリクエスト用に 1 回、このオブジェクトをキャッシュします。

> **Important**
>
> Amazon S3 および一部の HTTP サーバーは Cookie を処理しません。Cookie を処理しないオリジンや、Cookie に基づいてレスポンスを変化させないオリジンに Cookie を転送するように CloudFront を設定しないでください。これにより、CloudFront によって同じオブジェクトのオリジンに多くのリクエストが転送されるため、パフォーマンスが低下し、オリジンの負荷が増加する可能性があります。前の例を考慮すると、オリジンが `country` Cookie を処理しない場合や、`locations.html` Cookie の値に関係なく同じバージョンの `country` を CloudFront に返す場合は、その Cookie を転送するように CloudFront を設定しないでください。
>
> 逆に、カスタムオリジンが特定の Cookie に依存しているか、Cookie に基づいて異なるレスポンスを送信する場合は、その Cookie をオリジンに転送するように CloudFront を設定してください。そのように設定しない場合、CloudFront はリクエストをオリジンに転送する前に Cookie を削除します。

Cookie 転送を設定するには、ディストリビューションのキャッシュ動作を更新します。キャッシュ動作の詳細については、「キャッシュ動作の設定」の、特に「cookie の転送」および「許可リスト Cookie」セクションを参照してください。

各キャッシュ動作を設定して、次のいずれかを実行できます。

- オリジンにすべての Cookie を転送する - CloudFront には、リクエストをオリジンに転送するときにビューワーが送信するすべての Cookie が含まれます。オリジンがレスポンスを返すと、CloudFront はビューワーリクエスト内の Cookie の名前と値を使用してレスポンスをキャッシュします。オリジンレスポンスに `Set-Cookie` ヘッダーが含まれている場合、CloudFront はリクエストされたオブジェクトと共にそれらのヘッダーをビューワーに返します。CloudFront は、オリジンから返されたオブジェクトと共に `Set-Cookie` ヘッダーもキャッシュし、すべてのキャッシュヒットでそれらの `Set-Cookie` ヘッダーをビューワーに送信します。
- 指定した Cookie のセットを転送する - CloudFront は、リクエストをオリジンに転送する前に、アローリストにないビューワーが送信する Cookie をすべて削除します。CloudFront は、ビューワーリクエストのリストに登録された Cookie の名前と値を使用してレスポンスをキャッシュします。オリジンレスポンスに `Set-Cookie` ヘッダーが含まれている場合、CloudFront はリクエストされたオブジェクトと共にそれらのヘッダーをビューワーに返します。CloudFront は、オリジンから返されたオブジェクトと共に `Set-Cookie` ヘッダーもキャッシュし、すべてのキャッシュヒットでそれらの `Set-Cookie` ヘッダーをビューワーに送信します。

  Cookie 名でワイルドカードを指定する方法の詳細については、「許可リスト Cookie」を参照してください。

  キャッシュ動作ごとに転送できる Cookie 名の数に関する現在のクォータについて、またはクォータの引き上げをリクエストするには、「クエリ文字列のクォータ (従来のキャッシュ設定)」を参照してください。
- Cookie をオリジンに転送しない - CloudFront は、ビューワーから送信された Cookie に基づくオブジェクトをキャッシュしません。さらに、CloudFront はリクエストをオリジンに転送する前に Cookie を削除し、レスポンスをビューワーに返す前にレスポンスから `Set-Cookie` ヘッダーを削除します。これはオリジンリソースの最適な使用方法ではないため、このキャッシュビヘイビアを選択するときは、オリジンがデフォルトでオリジンレスポンスに Cookie を含めないようにする必要があります。

転送する Cookie を指定するときには、以下に注意してください。

アクセスログ

Cookie をオリジンに転送しないように CloudFront を設定した場合や、特定の Cookie のみを転送するように CloudFront を設定した場合でも、リクエストと Cookie を口グに記録するように CloudFront を設定すると、CloudFront ではすべての Cookie とすべての Cookie 属性が口グに記

録されます。CloudFront ログ記録の詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

大文字と小文字の区別

Cookie の名前と値は、大文字と小文字を区別します。例えば、CloudFront がすべての Cookie を転送するように設定されていて、同じオブジェクトに対する 2 つのビューワーリクエストに、大文字/小文字を除いて同一の Cookie がある場合、CloudFront はオブジェクトを 2 回キャッシュします。

CloudFront による Cookie の並べ替え

CloudFront が Cookie (すべてまたはサブセット) を転送するように設定されている場合、CloudFront はリクエストをオリジンに転送する前に、Cookie 名の自然な順序で Cookie を並べ替えます。

`If-Modified-Since` および `If-None-Match`

`If-Modified-Since` と `If-None-Match` の条件付きリクエストは、CloudFront が Cookie (すべてまたはサブセット) を転送するように設定されている場合はサポートされません。

標準の名前と値のペア形式が必要

CloudFront は、値が標準の名前と値のペア形式 (`"Cookie: cookie1=value1; cookie2=value2"` など) に準拠している場合のみ、Cookie ヘッダーを転送します。

`Set-Cookie` ヘッダーのキャッシュの無効化

CloudFront が Cookie をオリジン (すべてまたは特定の Cookie にかかわらず) に転送するように設定されている場合、オリジンレスポンスで受信した `Set-Cookie` ヘッダーもキャッシュします。CloudFront は、元のビューワーへのレスポンスにこれらの `Set-Cookie` ヘッダーを含め、CloudFront キャッシュから提供される後続のレスポンスにも含めます。

オリジンで Cookie を受信するが、CloudFront がオリジンのレスポンスで `Set-Cookie` ヘッダーをキャッシュしないようにする場合、フィールド名として `Cache-Control` を指定する `no-cache` ディレクティブを含む `Set-Cookie` ヘッダーを追加するようにオリジンを設定します。例: `Cache-Control: no-cache="Set-Cookie"`。詳細については、「Hypertext Transfer Protocol (HTTP/1.1): Caching」標準の「Response Cache-Control Directives」を参照してください。

Cookie 名の全体の長さ

特定の Cookie をオリジンに転送するように CloudFront を設定した場合、転送するように CloudFront を設定するすべての Cookie 名の合計バイト数は、512 から転送する Cookie の数

を引いた値を超えることはできません。例えば、10 個の Cookie をオリジンに転送するように CloudFront を設定する場合は、10 個の Cookie 名の合計の長さが 502 バイト (512 - 10) を超えないようにしてください。

すべての Cookie をオリジンに転送するように CloudFront を設定する場合は、Cookie 名の長さを考慮する必要はありません。

CloudFront コンソールを使用して、CloudFront で Cookie をオリジンに転送するようにディストリビューションを更新する方法については、「ディストリビューションを更新する」を参照してください。CloudFront API を使用してディストリビューションを更新する方法については、Amazon CloudFront API リファレンスの「UpdateDistribution」を参照してください。

リクエストヘッダーに基づいてコンテンツをキャッシュする

CloudFront により、ヘッダーをオリジンに転送し、ビューワーリクエストのヘッダー値に基づいて異なるバージョンのオブジェクトをキャッシュするかどうかを選択できます。こうすることで、ユーザーが使っているデバイスの種類やビューワーの場所、ビューワーで使われている言語など、さまざまな条件に基づいてコンテンツの異なるバージョンを配信できます。

トピック

- ヘッダーとディストリビューションの概要
- キャッシュ条件に使用するヘッダーを選択する
- CORS 設定を適用するように CloudFront を設定する
- デバイスタイプに基づいてキャッシュを設定する
- ビューワーの言語に基づいてキャッシュを設定する
- ビューワーの場所に基づいてキャッシュを設定する
- リクエストのプロトコルに基づいてキャッシュを設定する
- 圧縮ファイルのキャッシュを設定する
- ヘッダーに基づくキャッシュがパフォーマンスに及ぼす影響
- ヘッダーとヘッダー値の大文字小文字がキャッシュに及ぼす影響
- CloudFront がビューワーに返すヘッダー

ヘッダーとディストリビューションの概要

デフォルトで、CloudFront では、エッジロケーションでオブジェクトをキャッシュする際にヘッダーが考慮されません。オリジンが 2 つのオブジェクトを返し、そのリクエストヘッダーの値のみが異なる場合、CloudFront はオブジェクトの片方のみをキャッシュします。

ヘッダーをオリジンに転送するように CloudFront を設定できます。その場合、CloudFront は 1 つ以上のリクエストヘッダーの値に基づいてオブジェクトの複数のバージョンをキャッシュします。特定のヘッダーの値に基づいてオブジェクトをキャッシュするように CloudFront を設定するには、ディストリビューションのキャッシュ動作の設定を指定します。詳細については、「選択されたリクエストヘッダーに基づいたキャッシュ」を参照してください。

たとえば、`logo.jpg` のヘッダーオブジェクトがカスタム `Product` ヘッダーを含み、その値が `Acme` または `Apex` であるとします。`Product` ヘッダーの値に基づいてオブジェクトをキャッシュするように CloudFront を設定すると、CloudFront は `logo.jpg` に関するリクエストをオリジンに転送し、`Product` ヘッダーとヘッダー値を含めます。CloudFront は、`logo.jpg` ヘッダーの値が `Product` であるリクエスト用に 1 回、値が `Acme` であるリクエスト用に 1 回、`Apex` をキャッシュします。

ディストリビューションの各キャッシュ動作を以下のいずれかを実行するように設定できます。

- すべてのヘッダーをオリジンに転送する

> **Note**
>
> レガシーキャッシュ設定 - すべてのヘッダーをオリジンに転送するように CloudFront を設定した場合、CloudFront はこのキャッシュ動作に関連付けられたオブジェクトをキャッシュしません。その代わりに、すべてのリクエストをオリジンに送信します。

- 指定したヘッダーのリストを転送する。CloudFront は、指定されたヘッダーすべての値に基づいてオブジェクトをキャッシュします。CloudFront は、デフォルトで転送するヘッダーも転送しますが、指定されたヘッダーの値にのみ基づいてオブジェクトをキャッシュします。
- デフォルトのヘッダーのみを転送する。この設定の場合、CloudFront は、リクエストヘッダーの値に基づいてオブジェクトをキャッシュすることはありません。

キャッシュ動作ごとに転送できるヘッダーの数に関する現在のクォータについて、またはクォータの引き上げをリクエストするには、「ヘッダーのクォータ」を参照してください。

CloudFront コンソールを使用して、CloudFront でヘッダーをオリジンに転送するようにディストリビューションを更新する方法については、「ディストリビューションを更新する」を参照してください。CloudFront API を使用して既存のディストリビューションを更新する方法については、Amazon CloudFront API リファレンスの「UpdateDistribution」を参照してください。

キャッシュ条件に使用するヘッダーを選択する

オリジンに転送できるヘッダーおよび、CloudFront がキャッシュ条件にするヘッダーは、オリジンが Amazon S3 バケットであるか、カスタムオリジンを使用しているかに応じて変わります。

- Amazon S3 - 特定のヘッダーの数に基づいて、オブジェクトを転送し、キャッシュするように CloudFront を設定できます (以下の例外のリストを参照)。ただし、Cross-Origin Resource Sharing (CORS) を実装するか、オリジン側イベントで Lambda@Edge を使用してコンテンツをパーソナライズする必要がない限り、Amazon S3 オリジンを使用してヘッダーを転送しないようにすることをお勧めします。
  - CORS を設定するには、CloudFront に Cross-Origin Resource Sharing (CORS) が有効になっているウェブサイトのコンテンツの配信を許可するヘッダーを転送する必要があります。詳細については、「CORS 設定を適用するように CloudFront を設定する」を参照してください。
  - Amazon S3 オリジンに転送するヘッダーを使用してコンテンツをパーソナライズするには、Lambda@Edge 関数を記述および追加し、オリジン側イベントによってトリガーされる CloudFront ディストリビューションに関連付けます。ヘッダーの操作によるコンテンツのパーソナライズの詳細については、「国またはデバイスタイプヘッダー別のコンテンツのパーソナライズ - 例」を参照してください。

    使用していないヘッダーを転送してコンテンツをパーソナライズしないようにすることをお勧めします。追加のヘッダーを転送すると、キャッシュヒット率が低下する可能性があるためです。つまり、CloudFront はすべてのリクエストの比率として、エッジキャッシュからのすべてのリクエストに対応することができません。
- カスタムオリジン - 以下を除く任意のリクエストヘッダーの値に基づいてキャッシュするように CloudFront を設定できます。
  - `Connection`
  - `Cookie` - Cookie に基づいて転送しキャッシュする場合は、ディストリビューションの別の設定を使用します。詳細については、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。
  - `Host (for Amazon S3 origins)`
  - `Proxy-Authorization`

- TE
- Upgrade

Date および User-Agent ヘッダーの値に基づいてオブジェクトをキャッシュするように CloudFront を設定できますが、これはお勧めできません。これらのヘッダーには可能な値が多数あり、その値に基づいてキャッシュすると、CloudFront がオリジンに転送するリクエストの数が大幅に増加します。

すべての HTTP リクエストヘッダーの一覧と、CloudFront がそれを処理する方法については、「HTTP リクエストヘッダーと CloudFront の動作 (カスタムオリジンおよび Amazon S3 オリジン)」を参照してください。

CORS 設定を適用するように CloudFront を設定する

Cross-Origin Resource Sharing (CORS) を Amazon S3 バケットまたはカスタムオリジンで有効にしている場合、その CORS 設定を優先させるために、転送する特定のヘッダーを選択する必要があります。転送する必要があるヘッダーは、オリジン (Amazon S3 またはカスタム)、および OPTIONS レスポンスをキャッシュするかどうかによって異なります。

Amazon S3

- OPTIONS レスポンスをキャッシュする場合は、次の操作を行います。
  - OPTIONS レスポンスのキャッシュを有効にする、デフォルトのキャッシュ動作設定のオプションを選択します。
  - Origin、Access-Control-Request-Headers、および Access-Control-Request-Method ヘッダーを転送するように CloudFront を設定します。
- OPTIONS レスポンスをキャッシュしない場合は、オリジンに必要な他のヘッダー (Origin や Access-Control-Request-Headers など) と一緒に Access-Control-Request-Method ヘッダーを転送するように CloudFront を設定します。

カスタムオリジン - オリジンが必要とする他のヘッダーと共に、Origin ヘッダーを転送します。

CORS に基づいてレスポンスをキャッシュするように CloudFront を設定するには、キャッシュポリシーを使用してヘッダーを転送するように CloudFront を設定する必要があります。詳細については、「ポリシーを使用してキャッシュキーを制御する」を参照してください。

CORS と Amazon S3 の詳細については、Amazon Simple Storage Service ユーザーガイドの「Cross-Origin Resource Sharing (CORS) の使用」を参照してください。

デバイスタイプに基づいてキャッシュを設定する

ユーザーがコンテンツの表示に使用しているデバイスに基づいて、オブジェクトの異なるバージョンを CloudFront でキャッシュするには、該当するヘッダーをカスタムオリジンに転送するように CloudFront を設定します。

- `CloudFront-Is-Desktop-Viewer`
- `CloudFront-Is-Mobile-Viewer`
- `CloudFront-Is-SmartTV-Viewer`
- `CloudFront-Is-Tablet-Viewer`

CloudFront は、`User-Agent` ヘッダーの値に基づいて、これらのヘッダーの値を `true` または `false` に設定した後、リクエストをオリジンに転送します。デバイスが複数のカテゴリに属する場合は、複数の値が `true` になることがあります。たとえば、一部のタブレットデバイスに対して、CloudFront が `CloudFront-Is-Mobile-Viewer` と `CloudFront-Is-Tablet-Viewer` の両方を `true` に設定する場合があります。

ビューワーの言語に基づいてキャッシュを設定する

リクエストで指定された言語に基づいて、オブジェクトの異なるバージョンを CloudFront でキャッシュするには、`Accept-Language` ヘッダーをオリジンに転送するように CloudFront を設定します。

ビューワーの場所に基づいてキャッシュを設定する

リクエスト元の国に基づいて、オブジェクトの異なるバージョンを CloudFront でキャッシュするには、`CloudFront-Viewer-Country` ヘッダーをオリジンに転送するように CloudFront を設定します。CloudFront はリクエスト元の IP アドレスを 2 文字の国コードに自動的に変換します。コード順、国順に並べ替えることのできる使いやすい国コードの一覧については、Wikipedia の「ISO 3166-1 alpha-2」の項目を参照してください。

リクエストのプロトコルに基づいてキャッシュを設定する

リクエストのプロトコル (HTTP または HTTPS) に基づいて、オブジェクトの異なるバージョンを CloudFront でキャッシュするには、CloudFront-Forwarded-Proto ヘッダーをオリジンに転送するように CloudFront を設定します。

圧縮ファイルのキャッシュを設定する

オリジンが Brotli 圧縮をサポートしている場合は、Accept-Encoding ヘッダーに基づいてキャッシュできます。オリジンがヘッダーに基づいて異なるコンテンツを配信する場合のみ、Accept-Encoding に基づいてキャッシュを設定する必要があります。

ヘッダーに基づくキャッシュがパフォーマンスに及ぼす影響

ヘッダーに基づいてキャッシュするように CloudFront を設定した場合、ヘッダーで指定できる値が複数あると、CloudFront が同じオブジェクトについてオリジンサーバーに転送するリクエストの数が増えます。このためパフォーマンスが低下し、オリジンサーバーの負荷が増加します。所定のヘッダーの値に関係なくオリジンサーバーが同じオブジェクトを返す場合は、そのヘッダーに基づいてキャッシュするように CloudFront を設定しないことをお勧めします。

複数のヘッダーを転送するように CloudFront を設定した場合、ビューワーリクエストのヘッダーの順序は、値が同じである限り、キャッシュ動作には影響しません。例えば、あるリクエストのヘッダーが A:1、B:2 で、別のリクエストのヘッダーが B:2、A:1 である場合、CloudFront はそのオブジェクトのコピーを 1 つだけキャッシュします。

ヘッダーとヘッダー値の大文字小文字がキャッシュに及ぼす影響

CloudFront がヘッダー値に基づいてキャッシュする場合、ヘッダー名の大文字小文字の違いは無視されますが、ヘッダー値の大文字小文字の違いは考慮されます。

- ビューワーリクエストが Product:Acme と product:Acme の両方を含む場合、CloudFront がオブジェクトをキャッシュするのは 1 回だけです。両者の違いはヘッダー名の大文字小文字だけで、これはキャッシュ動作に影響しません。
- ビューワーリクエストが Product:Acme と Product:acme の両方を含む場合、CloudFront はオブジェクトを 2 回キャッシュします。値が、あるリクエストでは Acme、別のリクエストでは acme と異なっているためです。

CloudFront がビューワーに返すヘッダー

ヘッダーを転送およびキャッシュするように CloudFront を設定しても、CloudFront がビューワーに返すヘッダーには影響しません。CloudFront は、いくつかの例外を除いて、オリジンから取得したヘッダーをすべて返します。詳細については、該当するトピックを参照してください。

- Amazon S3 のオリジン - 「CloudFront が削除または更新する HTTP レスポンスヘッダー」を参照してください。
- カスタムオリジン - 「CloudFront が削除または置き換える HTTP レスポンスヘッダー」を参照してください。

ポリシーを使用してキャッシュキーを制御する

CloudFront のキャッシュポリシーを使用して、CloudFront エッジロケーションにキャッシュされるオブジェクトのキャッシュキーに CloudFront が含める HTTP ヘッダー、Cookie、およびクエリ文字列を指定できます。キャッシュキーは、キャッシュ内のオブジェクトごとの一意の識別子であり、ビューワーの HTTP リクエストがキャッシュヒットになるかどうかを決定します。

キャッシュヒットが発生するのは、ビューワーリクエストが以前のリクエストと同じキャッシュキーを生成し、そのキャッシュキーのオブジェクトがエッジロケーションのキャッシュにあり、有効な場合です。キャッシュヒットが発生すると、オブジェクトが CloudFront エッジロケーションからビューワーに提供されます。これには以下の利点があります。

- オリジンサーバーの負荷を軽減
- ビューワーのレイテンシーを低減

キャッシュキーに含まれる値が少なくなると、キャッシュヒットの可能性が高まります。キャッシュヒット率が高い (キャッシュヒットになるビューワーリクエストの割合が高い) ほど、ウェブサイトやアプリケーションのパフォーマンスが高くなります。詳細については、「キャッシュキーを理解する」を参照してください。

キャッシュキーを管理するには、CloudFront キャッシュポリシーを使用します。キャッシュポリシーは、CloudFront ディストリビューションの 1 つ以上のキャッシュ動作にアタッチします。

また、キャッシュポリシーを使用して CloudFront キャッシュ内のオブジェクトの有効期限 (TTL) 設定が指定でき、CloudFront での圧縮オブジェクトのリクエストおよびキャッシュができるようになります。

トピック

キャッシュポリシーを理解する

キャッシュポリシーを使用して、キャッシュキーに含まれる値 (URL クエリ文字列、HTTP ヘッダー、Cookie) を管理することで、キャッシュヒット率を改善できます。CloudFront には、一般的なユースケース用にマネージドポリシーと呼ばれる事前定義されたオリジンリクエストポリシーがいくつか用意されています。これらのマネージドポリシーを使用することも、ユーザーのニーズ別に独自のキャッシュポリシーを作成することもできます。マネージドポリシーの詳細については、「マネージドキャッシュポリシーを使用する」を参照してください。

キャッシュポリシーには以下の設定が含まれます。設定は、ポリシー情報、Time to live (TTL) 設定、およびキャッシュキー設定に分類されます。

ポリシー情報

名前

キャッシュポリシーを特定する名前。コンソールでは、名前を使用して、キャッシュポリシーをキャッシュ動作にアタッチします。

説明

キャッシュポリシーを説明するコメント。これはオプションですが、キャッシュポリシーの目的を特定するのに役立ちます。

Time to live (TTL) 設定

Time to live (TTL) 設定は、`Cache-Control` および `Expires` HTTP ヘッダー (オリジンレスポンス内にある場合) と連動して、CloudFront キャッシュ内のオブジェクトの有効期間を決定します。

最小 TTL

CloudFront がオリジンをチェックしてオブジェクトが更新されているかどうかを確認するまでに、オブジェクトが CloudFront キャッシュに保持される最小期間 (秒単位)。詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

最大 TTL

CloudFront がオリジンをチェックしてオブジェクトが更新されているかどうかを確認するまでに、オブジェクトが CloudFront キャッシュに保持される最大期間 (秒単位)。CloudFront は、オリジンがオブジェクトと共に `Cache-Control` または `Expires` ヘッダーを送信する場合のみ、

この設定を使用します。詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

デフォルト TTL

CloudFront がオリジンをチェックしてオブジェクトが更新されているかどうかを確認するまでに、オブジェクトを CloudFront キャッシュに保持するデフォルトの時間 (秒単位)。CloudFront は、オリジンがオブジェクトと共に `Cache-Control` ヘッダーまたは `Expires` ヘッダーを送信しない場合にのみ、この設定の値をオブジェクトの TTL として使用します。詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

> **Note**
>
> [最小 TTL]、[最大 TTL]、[デフォルト TTL] の設定がすべて 0 に設定されている場合、CloudFront のキャッシュは無効になります。

キャッシュキー設定

キャッシュキーの設定では、CloudFront によりキャッシュキーに含まれるビューワーリクエストの値を指定します。値には、URL クエリ文字列、HTTP ヘッダー、および Cookie を含めることができます。キャッシュキーに含める値は、CloudFront がオリジンに送信するリクエスト (オリジンリクエストと呼ばれる) に自動的に含まれます。キャッシュキーに影響を与えずにオリジンリクエストを管理する方法については、「ポリシーを使用してオリジンリクエストを制御する」を参照してください。

キャッシュキーには次のような設定があります。

- ヘッダー
- Cookie
- クエリ文字列
- 圧縮のサポート

ヘッダー

CloudFront によりキャッシュキーおよびオリジンリクエストに含まれる、ビューワーリクエストの HTTP ヘッダー。ヘッダーには、以下のいずれかの設定を選択できます。

- [None (なし)] - ビューワーリクエストの HTTP ヘッダーはキャッシュキーに含まれず、オリジンリクエストに自動的に含まれません。
- [次のヘッダーを含める] - ビューワーリクエストのどの HTTP ヘッダーをキャッシュキーに含め、オリジンリクエストに自動的に含めるかを指定します。

[次のヘッダーを含める] 設定を使用する場合、HTTP ヘッダーは値ではなく名前で指定します。たとえば、次の HTTP ヘッダーを考えてみます。

```
Accept-Language: en-US,en;q=0.5
```

この場合、ヘッダーを `Accept-Language: en-US,en;q=0.5` としてではなく、`Accept-Language` として指定します。ただし、CloudFront では、キャッシュキーおよびオリジンリクエストに、その値を含む完全なヘッダーが含まれます。

CloudFront によって生成された特定のヘッダーをキャッシュキーに含めることもできます。詳細については、「the section called “CloudFront のリクエストヘッダーを追加する”」を参照してください。

Cookie

CloudFront によりキャッシュキーおよびオリジンリクエストに含まれる、ビューワーリクエストの Cookie。Cookie には、以下のいずれかの設定を選択できます。

- [None (なし)] - ビューワーリクエストの Cookie はキャッシュキーに含まれず、オリジンリクエストに自動的に含まれません。
- [All (すべて)] - ビューワーリクエストのすべての Cookie は、キャッシュキーに含まれ、オリジンリクエストに自動的に含まれます。
- [指定した Cookie を含める] - ビューワーリクエストのどの Cookie をキャッシュキーに含め、オリジンリクエストに自動的に含めるかを指定します。
- [次を除くすべての Cookie を含める] - ビューワーリクエストのどの Cookie をキャッシュキーに含めず、オリジンリクエストに自動的に含めないかを指定します。指定した Cookie 以外の他のすべての Cookie は、キャッシュキーに含まれ、自動的にオリジンリクエストに含まれます。

[指定した Cookie を含める] または [次を除くすべての Cookie を含める] 設定を使用する場合、Cookie は値ではなく名前で指定します。たとえば、次の `Cookie` ヘッダーを考えてみます。

```
Cookie: session_ID=abcd1234
```

この場合、Cookie を session_ID=abcd1234 としてではなく、session_ID として指定します。ただし、CloudFront では、キャッシュキーおよびオリジンリクエストに、その値を含む完全な Cookie が含まれます。

クエリ文字列

CloudFront によりキャッシュキーおよびオリジンリクエストに含まれる、ビューワーリクエストの URL クエリ文字列。クエリ文字列には、以下のいずれかの設定を選択できます。

- [None (なし)] - ビューワーリクエストのクエリ文字列はキャッシュキーに含まれず、オリジンリクエストに自動的に含まれません。
- [All (すべて)] - ビューワーリクエストのすべてのクエリ文字列は、キャッシュキーに含まれ、オリジンリクエストにも自動的に含まれます。
- [指定したクエリ文字列を含める] - ビューワーリクエストのどのクエリ文字列をキャッシュキーに含め、オリジンリクエストに自動的に含めるかを指定します。
- [次を除くすべてのクエリ文字列を含める] - ビューワーリクエストのどのクエリ文字列をキャッシュキーに含めず、オリジンリクエストに自動的に含めないかを指定します。指定したクエリ文字列以外の他のすべてのクエリ文字列は、キャッシュキーに含まれ、自動的にオリジンリクエストに含まれます。

[指定したクエリ文字列を含める] または [次を除くすべてのクエリ文字列を含める] 設定を使用する場合、クエリ文字列は値ではなく名前で指定します。たとえば、次の URL パスを考えてみます。

```
/content/stories/example-story.html?split-pages=false
```

この場合、クエリ文字列を split-pages=false としてではなく、split-pages として指定します。ただし、CloudFront では、キャッシュキーおよびオリジンリクエストに、その値を含む完全なクエリ文字列が含まれます。

圧縮のサポート

これらの設定により、Gzip または Brotli 圧縮形式で圧縮されたオブジェクトを CloudFront がリクエストおよびキャッシュできるようになります (ビューワーでサポートされている場合)。これらの設定により、CloudFront 圧縮も有効になります。ビューワーは Accept-Encoding HTTP ヘッダーを使用して、これらの圧縮形式のサポートの可否を示します。

> **Note**
>
> ウェブブラウザ Chrome および Firefox では、HTTPS を使用してリクエストを送信する場合のみ、Brotli 圧縮がサポートされます。これらのブラウザでは、HTTP リクエストで Brotli がサポートされません。

次のいずれかに該当する場合は、これらの設定を有効にします。

- ビューワーによってサポートされている場合にオリジンが Gzip 圧縮オブジェクトを返す (リクエストには、HTTPヘッダー `Accept-Encoding` と値 `gzip` が含まれます)。この場合、Gzip が有効化された設定を使用します (CloudFront API、AWS SDK、AWS CLI、または AWS CloudFormation で `EnableAcceptEncodingGzip` を `true` に設定します)。
- ビューワーによってサポートされている場合にオリジンが Brotli 圧縮オブジェクトを返す (リクエストには、HTTPヘッダー `Accept-Encoding` と値 `br` が含まれます)。この場合、Brotli が有効化された設定を使用します (CloudFront API、AWS SDK、AWS CLI、または AWS CloudFormation で `EnableAcceptEncodingBrotli` を `true` に設定します)。
- このキャッシュポリシーが適用されるキャッシュ動作は、CloudFront 圧縮によって設定されます。この場合、Gzip または Brotli のいずれか、またはその両方に対してキャッシュを有効にできます。CloudFront 圧縮が有効になっている場合、両方の形式でキャッシュを有効にすると、インターネットへのデータ転送のコストを軽減できます。

> **Note**
>
> これらの圧縮形式の一方または両方でキャッシュを有効にする場合は、同じキャッシュ動作に関連付けられているオリジンリクエストポリシーに `Accept-Encoding` ヘッダーを含めないでください。CloudFront は、これらの形式のいずれかでキャッシュが有効になっている場合、常にこのヘッダーをオリジンリクエストに含めます。そのため、オリジンリクエストポリシーに `Accept-Encoding` を含めても効果はありません。

オリジンサーバーが Gzip または Brotli で圧縮されたオブジェクトを返さない場合、またはキャッシュ動作が CloudFront 圧縮で設定されていない場合は、圧縮オブジェクトのキャッシュを有効にしないでください。有効にすると、キャッシュヒット率が低下する可能性があります。

以下に、これらの設定が CloudFront ディストリビューションに与える影響について説明します。次のシナリオはすべて、ビューワーリクエストに `Accept-Encoding` ヘッダーが含まれていることを前提としています。ビューワーリクエストに `Accept-Encoding` ヘッダーが含まれてい

ない場合、CloudFront はこのヘッダーをキャッシュキーに含めず、対応するオリジンリクエストにも含めません。

圧縮されたオブジェクトのキャッシュが両方の圧縮形式に対応している場合

ビューワーが Gzip と Brotli の両方をサポートしている場合、つまり、ビューワーリクエストの `Accept-Encoding` ヘッダーに `gzip` 値と `br` 値の両方が含まれている場合、CloudFront では以下の処理が行われます。

- ヘッダーを `Accept-Encoding: br,gzip` の形式に正規化し、正規化されたヘッダーをキャッシュキーに含めます。キャッシュキーには、ビューワーから送信された `Accept-Encoding` ヘッダーに存在していた他の値は含まれません。
- エッジロケーションのキャッシュに、リクエストに一致し有効期限が切れていない Brotli または Gzip 圧縮オブジェクトがある場合、エッジロケーションはそのオブジェクトをビューワーに返します。
- エッジロケーションのキャッシュに、リクエストと一致し、有効期限が切れていない Brotli または Gzip 圧縮オブジェクトがない場合、CloudFront は対応するオリジンリクエストに正規化されたヘッダー (`Accept-Encoding: br,gzip`) を含めます。オリジンリクエストには、ビューワーから送信された `Accept-Encoding` ヘッダーに存在していた他の値は含まれません。

例えば、ビューワーが一方の圧縮形式をサポートし、他方の圧縮形式をサポートしていない場合、ビューワーリクエストの `Accept-Encoding` ヘッダーに値 `gzip` が指定され、`br` が指定されていない場合、CloudFront によって以下の処理が行われます。

- ヘッダーを `Accept-Encoding: gzip` の形式に正規化し、正規化されたヘッダーをキャッシュキーに含めます。キャッシュキーには、ビューワーから送信された `Accept-Encoding` ヘッダーに存在していた他の値は含まれません。
- エッジロケーションのキャッシュに、リクエストに一致し有効期限が切れていない Gzip 圧縮オブジェクトがある場合、エッジロケーションはそのオブジェクトをビューワーに返します。
- エッジロケーションのキャッシュに、リクエストと一致し、有効期限が切れていない Gzip 圧縮オブジェクトがない場合、CloudFront は対応するオリジンリクエストに正規化されたヘッダー (`Accept-Encoding: gzip`) を含めます。オリジンリクエストには、ビューワーから送信された `Accept-Encoding` ヘッダーに存在していた他の値は含まれません。

ビューワーが Brotli をサポートし Gzip をサポートしていない場合の CloudFront の動作を理解するには、前の例で圧縮形式を他方に置き換えてください。

ビューワーが Brotli または Gzip をサポートしていない場合、つまりビューワーリクエストの Accept-Encoding ヘッダーに br または gzip の値が含まれていない場合、CloudFront の動作は以下のようになります。

- キャッシュキーに Accept-Encoding ヘッダーを含めません。
- 対応するオリジンリクエストに Accept-Encoding: identity を含めます。オリジンリクエストには、ビューワーから送信された Accept-Encoding ヘッダーに存在していた他の値は含まれません。

圧縮されたオブジェクトのキャッシュが 1 つの圧縮形式に対して有効で、もう 1 つの圧縮形式に対して有効でない場合

キャッシュが対応している形式をビューワーがサポートしている場合、例えば、圧縮されたオブジェクトのキャッシュが Gzip に対応していて、ビューワーが Gzip をサポートしている (ビューワーリクエストの Accept-Encoding ヘッダー内の値の 1 つが gzip である) 場合、CloudFront によって以下の処理が行われます。

- ヘッダーを Accept-Encoding: gzip の形式に正規化し、正規化されたヘッダーをキャッシュキーに含めます。
- エッジロケーションのキャッシュに、リクエストに一致し有効期限が切れていない Gzip 圧縮オブジェクトがある場合、エッジロケーションはそのオブジェクトをビューワーに返します。
- エッジロケーションのキャッシュに、リクエストと一致し、有効期限が切れていない Gzip 圧縮オブジェクトがない場合、CloudFront は対応するオリジンリクエストに正規化されたヘッダー (Accept-Encoding: gzip) を含めます。オリジンリクエストには、ビューワーから送信された Accept-Encoding ヘッダーに存在していた他の値は含まれません。

この動作は、ビューワーが Gzip と Brotli の両方をサポートしている場合と同じです (ビューワーリクエストの Accept-Encoding ヘッダーには gzip と br の両方の値が含まれます)。このシナリオでは、圧縮オブジェクトのキャッシュが Brotli に対応していないためです。

圧縮オブジェクトのキャッシュが Brotli に対応し Gzip に対応していない場合の CloudFront の動作を理解するには、前の例で圧縮形式を他方に置き換えてください。

キャッシュが対応している圧縮形式をビューワーがサポートしていない (ビューワーリクエストの Accept-Encoding ヘッダーに、その形式の値が含まれていない) 場合、CloudFront の動作は以下のようになります。

- キャッシュキーに Accept-Encoding ヘッダーを含めません。

- 対応するオリジンリクエストに `Accept-Encoding: identity` を含めます。オリジンリクエストには、ビューワーから送信された `Accept-Encoding` ヘッダーに存在していた他の値は含まれません。

圧縮されたオブジェクトのキャッシュが両方の圧縮形式に対応していない場合

圧縮オブジェクトのキャッシュが両方の圧縮形式に対応していない場合、CloudFront は、`Accept-Encoding` ヘッダーをビューワーリクエストの他の HTTP ヘッダーと同じように扱います。デフォルトでは、キャッシュキーには含まれず、オリジンリクエストにも含まれません。他の HTTP ヘッダーと同一のキャッシュポリシーまたはオリジンリクエストポリシーのヘッダーリストに含めることができます。

キャッシュポリシーを作成する

キャッシュポリシーを使用して、キャッシュキーに含まれる値 (URL クエリ文字列、HTTP ヘッダー、Cookie) を管理することで、キャッシュヒット率を改善できます。キャッシュポリシーは、CloudFront コンソール、AWS Command Line Interface (AWS CLI)、または CloudFront API を使用して作成できます。

キャッシュポリシーを作成したら、CloudFront ディストリビューションの 1 つ以上のキャッシュ動作にアタッチします。

Console

キャッシュポリシーを作成するには (コンソール)

1. AWS Management Console にログインし、https://console.aws.amazon.com/cloudfront/v4/home?#/policies で CloudFront コンソールの [Policies] (ポリシー) ページを開きます。
2. [キャッシュポリシーの作成] を選択します。
3. このキャッシュポリシーに目的の設定を選択します。詳細については、「キャッシュポリシーを理解する」を参照してください。
4. 終了したら、[作成] を選択します。

キャッシュポリシーを作成したら、それをキャッシュ動作にアタッチできます。

既存のディストリビューションにキャッシュポリシーをアタッチするには (コンソール)

1. https://console.aws.amazon.com/cloudfront/v4/home#/distributions で CloudFront コンソールの [Distributions] (ディストリビューション) ページを開きます。

2. 更新するディストリビューションを選択し、[動作] タブを選択します。
3. 更新するキャッシュ動作を選択し、[編集] を選択します。

   または、新しいキャッシュ動作を作成するには、[動作を作成] を選択します。
4. [キャッシュキーとオリジンリクエスト] セクションで、[キャッシュポリシーとオリジンリクエストポリシー] が選択されていることを確認します。
5. [キャッシュポリシー] では、このキャッシュ動作にアタッチするキャッシュポリシーを選択します。
6. ページの最下部で [変更の保存] を選択します。

新しいディストリビューションにキャッシュポリシーをアタッチするには (コンソール)

1. で CloudFront コンソールを開きますhttps://console.aws.amazon.com/cloudfront/v4/home
2. [ディストリビューションの作成] を選択します。
3. [キャッシュキーとオリジンリクエスト] セクションで、[キャッシュポリシーとオリジンリクエストポリシー] が選択されていることを確認します。
4. [Cache policy] (キャッシュポリシー) で、このディストリビューションのデフォルトのキャッシュ動作にアタッチするキャッシュポリシーを選択します。
5. オリジン、デフォルトのキャッシュ動作、その他のディストリビューション設定に必要な設定を選択します。詳細については、「ディストリビューション設定リファレンス」を参照してください。
6. 終了したら、[ディストリビューションの作成] を選択します。

CLI

AWS Command Line Interface (AWS CLI) でキャッシュポリシーを作成するには、aws cloudfront create-cache-policy コマンドを使用します。コマンドの入力パラメータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

キャッシュポリシーを作成するには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、`cache-policy.yaml` コマンドのすべての入力パラメータを含む create-cache-policy という名前のファイルを作成します。

```
aws cloudfront create-cache-policy --generate-cli-skeleton yaml-input > cache-
policy.yaml
```

2. 先ほど作成した cache-policy.yaml という名前のファイルを開きます。ファイルを編集して、必要なキャッシュポリシー設定を指定し、ファイルを保存します。ファイルからオプションのフィールドを削除することはできますが、必須フィールドは削除しないでください。

   キャッシュポリシー設定の詳細については、「キャッシュポリシーを理解する」を参照してください。

3. 次のコマンドを使用して、cache-policy.yaml ファイルの入力パラメータを使用し、キャッシュポリシーを作成します。

   ```
   aws cloudfront create-cache-policy --cli-input-yaml file://cache-policy.yaml
   ```

   コマンドの出力の Id 値を書き留めます。これはキャッシュポリシー ID であり、キャッシュポリシーを CloudFront ディストリビューションのキャッシュ動作にアタッチするために必要です。

既存のディストリビューションにキャッシュポリシーをアタッチするには (入力ファイルを含む CLI)

1. 以下のコマンドを使用して、更新する CloudFront ディストリビューションのディストリビューション設定を保存します。*distribution_ID* をディストリビューションの ID に置き換えます。

   ```
   aws cloudfront get-distribution-config --id distribution_ID --output yaml >
    dist-config.yaml
   ```

2. 先ほど作成した dist-config.yaml という名前のファイルを開きます。ファイルを編集し、キャッシュポリシーを使用するように更新する各キャッシュ動作に次の変更を加えます。

   - キャッシュ動作で、CachePolicyId という名前のフィールドを追加します。フィールドの値には、ポリシーの作成後に書き留めたキャッシュポリシー ID を使用します。

- キャッシュ動作から MinTTL、MaxTTL、DefaultTTL、および ForwardedValues フィールドを削除します。これらの設定はキャッシュポリシー内に指定するため、これらのフィールドとキャッシュポリシーを同じキャッシュ動作に含めることはできません。
- ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. キャッシュポリシーを使用するようにディストリビューションを更新するには、次のコマンドを使用します。*distribution_ID* をディストリビューションの ID に置き換えます。

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

新しいディストリビューションにキャッシュポリシーをアタッチするには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、distribution.yaml コマンドのすべての入力パラメータを含む create-distribution という名前のファイルを作成します。

```
aws cloudfront create-distribution --generate-cli-skeleton yaml-input >
 distribution.yaml
```

2. 先ほど作成した distribution.yaml という名前のファイルを開きます。デフォルトのキャッシュ動作の [CachePolicyId] フィールドに、ポリシーの作成後に書き留めたキャッシュポリシー ID を入力します。ファイルの編集を続行して必要なディストリビューション設定を指定し、完了したらファイルを保存します。

   ディストリビューション設定の詳細については、「ディストリビューション設定リファレンス」を参照してください。

3. 次のコマンドを使用して、distribution.yaml ファイルの入力パラメータを使用し、ディストリビューションを作成します。

```
aws cloudfront create-distribution --cli-input-yaml file://distribution.yaml
```

API

CloudFront API を使用してキャッシュポリシーを作成するには、CreateCachePolicy を使用します。この API コールで指定するフィールドの詳細については、「キャッシュポリシーを理解する」、および AWS SDK やその他の API クライアントの API リファレンスドキュメントを参照してください。

キャッシュポリシーを作成したら、次の API コールのいずれかを使用して、それをキャッシュ動作にアタッチできます。

- 既存のディストリビューションのキャッシュ動作にアタッチするには、UpdateDistribution を使用します。
- 新しいディストリビューションのキャッシュ動作にアタッチするには、CreateDistribution を使用します。

これらの API コールの両方について、キャッシュ動作内で、`CachePolicyId` フィールドにキャッシュポリシーの ID を指定します。これらの API コールで指定するその他フィールドの詳細については、「ディストリビューション設定リファレンス」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

マネージドキャッシュポリシーを使用する

CloudFront には、ディストリビューションのキャッシュ動作にアタッチできる一連のマネージドキャッシュポリシーが用意されています。マネージドキャッシュポリシーを使用すると、独自のキャッシュポリシーを記述したり、維持したりする必要はありません。マネージドポリシーでは、特定のユースケースに最適化された設定を使用します。

マネージドキャッシュポリシーを使用するには、ディストリビューションのキャッシュ動作にそのポリシーをアタッチします。このプロセスは、キャッシュポリシーを作成するときと同じですが、新しいキャッシュポリシーを作成するのではなく、マネージドキャッシュポリシーの 1 つをアタッチするだけです。ポリシーをアタッチするには、名前 (コンソールの場合) または ID (AWS CLI または SDK の場合) を使用します。名前と ID は、次のセクションに記載されています。

詳細については、「キャッシュポリシーを作成する」を参照してください。

次のトピックでは、使用可能なマネージドキャッシュポリシーについて説明します。

トピック

- Amplify
- CachingDisabled
- CachingOptimized
- CachingOptimizedForUncompressedObjects
- Elemental-MediaPackage
- UseOriginCacheControlHeaders
- UseOriginCacheControlHeaders-QueryStrings

Amplify

このポリシーを CloudFront コンソールで見る

このポリシーは、AWS Amplify ウェブアプリケーションであるオリジンで使用するように設計されています。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
2e54312d-136d-493c-8eb9-b001f22f67d2
```

このポリシーの設定は以下のとおりです。

- 最小 TTL: 2 秒
- 最大 TTL: 600 秒 (10 分)
- デフォルト TTL: 2 秒
- キャッシュキーに含まれるヘッダー:
  - `Authorization`
  - `CloudFront-Viewer-Country`
  - `Host`

  圧縮オブジェクトのキャッシュの設定が有効になっているため、正規化された `Accept-Encoding` ヘッダーも含まれます。詳細については、「圧縮のサポート」を参照してください。
- キャッシュキーに含まれる Cookie: すべての Cookie が含まれます。
- キャッシュキーに含まれるクエリ文字列: すべてのクエリ文字列が含まれます。

- 圧縮オブジェクトのキャッシュの設定: 有効。詳細については、「圧縮のサポート」を参照してください。

CachingDisabled

このポリシーを CloudFront コンソールで見る

このポリシーは、キャッシュを無効にします。このポリシーは、動的コンテンツとキャッシュできないリクエストに役立ちます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
4135ea2d-6df8-44a3-9df3-4b5a84be39ad
```

このポリシーの設定は以下のとおりです。

- 最小 TTL: 0 秒
- 最大 TTL: 0 秒
- デフォルト TTL: 0 秒
- キャッシュキーに含まれるヘッダー: なし
- キャッシュキーに含まれる Cookie: なし
- キャッシュキーに含まれるクエリ文字列: なし
- 圧縮オブジェクトのキャッシュの設定: 無効

CachingOptimized

このポリシーを CloudFront コンソールで見る

このポリシーは、CloudFront がキャッシュキーに含める値を最小限に抑えることで、キャッシュ効率を最適化するよう設計されています。CloudFront は、キャッシュキーにクエリ文字列や Cookie を含めず、正規化された `Accept-Encoding` ヘッダーのみを含めます。これにより CloudFront は、オリジンがオブジェクトを返すときまたは CloudFront エッジ圧縮が有効になっているときに、Gzip 圧縮形式と Brotli 圧縮形式でオブジェクトを個別にキャッシュできます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
658327ea-f89d-4fab-a63d-7e88639e58f6
```

このポリシーの設定は以下のとおりです。

- 最小 TTL: 1 秒。
- 最大 TTL: 31,536,000 秒 (365 日)。
- デフォルト TTL: 86,400 秒 (24 時間)。
- キャッシュキーに含まれるヘッダー: 明示的に含まれているものがありません。圧縮オブジェクトのキャッシュの設定が有効になっているため、正規化された `Accept-Encoding` ヘッダーが含まれます。詳細については、「圧縮のサポート」を参照してください。
- キャッシュキーに含まれる Cookie: なし。
- キャッシュキーに含まれるクエリ文字列: なし。
- 圧縮オブジェクトのキャッシュの設定: 有効。詳細については、「圧縮のサポート」を参照してください。

CachingOptimizedForUncompressedObjects

このポリシーを CloudFront コンソールで見る

このポリシーは、キャッシュキーに含まれる値を最小限に抑えることで、キャッシュ効率を最適化するよう設計されています。クエリ文字列、ヘッダー、または Cookie は含まれません。このポリシーは、前のポリシーと同じですが、圧縮オブジェクトのキャッシュの設定が無効になります。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
b2884449-e4de-46a7-ac36-70bc7f1ddd6d
```

このポリシーの設定は以下のとおりです。

- 最小 TTL: 1 秒
- 最大 TTL: 31,536,000 秒 (365 日)
- デフォルト TTL: 86,400 秒 (24 時間)
- キャッシュキーに含まれるヘッダー: なし
- キャッシュキーに含まれる Cookie: なし
- キャッシュキーに含まれるクエリ文字列: なし
- 圧縮オブジェクトのキャッシュの設定: 無効

Elemental-MediaPackage

このポリシーを CloudFront コンソールで見る

このポリシーは、オリジンとしての AWS Elemental MediaPackage 用に設計されています。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

`08627262-05a9-4f76-9ded-b50ca2e3a84f`

このポリシーの設定は以下のとおりです。

- 最小 TTL: 0 秒
- 最大 TTL: 31,536,000 秒 (365 日)
- デフォルト TTL: 86,400 秒 (24 時間)
- キャッシュキーに含まれるヘッダー:
  - `Origin`

  圧縮オブジェクトのキャッシュの設定が Gzip に対して有効になっているため、正規化された `Accept-Encoding` ヘッダーも含まれます。詳細については、「圧縮のサポート」を参照してください。
- キャッシュキーに含まれる Cookie: なし
- キャッシュキーに含まれるクエリ文字列:
  - `aws.manifestfilter`
  - `start`
  - `end`
  - `m`
- 圧縮オブジェクトのキャッシュの設定: Gzip に対して有効にする。詳細については、「圧縮のサポート」を参照してください。

UseOriginCacheControlHeaders

このポリシーを CloudFront コンソールで見る

このポリシーは、`Cache-Control` HTTP レスポンスヘッダーを返し、クエリ文字列内の値に応じて異なるコンテンツを提供しないオリジン用に設計されています。クエリ文字列内の値に応じて異なる

コンテンツを提供するオリジンの場合は、UseOriginCacheControlHeaders-QueryStrings の使用を検討してください。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
83da9c7e-98b4-4e11-a168-04f0df8e2c65
```

このポリシーの設定は以下のとおりです。

- 最小 TTL: 0 秒
- 最大 TTL: 31,536,000 秒 (365 日)
- デフォルト TTL: 0 秒
- キャッシュキーに含まれるヘッダー:
  - `Host`
  - `Origin`
  - `X-HTTP-Method-Override`
  - `X-HTTP-Method`
  - `X-Method-Override`

  圧縮オブジェクトのキャッシュの設定が有効になっているため、正規化された `Accept-Encoding` ヘッダーも含まれます。詳細については、「圧縮のサポート」を参照してください。
- キャッシュキーに含まれる Cookie: すべての Cookie が含まれます。
- キャッシュキーに含まれるクエリ文字列: なし。
- 圧縮オブジェクトのキャッシュの設定: 有効。詳細については、「圧縮のサポート」を参照してください。

UseOriginCacheControlHeaders-QueryStrings

このポリシーを CloudFront コンソールで見る

このポリシーは、`Cache-Control` HTTP レスポンスヘッダーを返し、クエリ文字列内の値に応じて異なるコンテンツを提供するオリジン用に設計されています。クエリ文字列内の値に応じて異なるコンテンツを提供しないオリジンの場合は、UseOriginCacheControlHeaders の使用を検討してください。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

`4cc15a8a-d715-48a4-82b8-cc0b614638fe`

このポリシーの設定は以下のとおりです。

- 最小 TTL: 0 秒
- 最大 TTL: 31,536,000 秒 (365 日)
- デフォルト TTL: 0 秒
- キャッシュキーに含まれるヘッダー:
  - `Host`
  - `Origin`
  - `X-HTTP-Method-Override`
  - `X-HTTP-Method`
  - `X-Method-Override`

  圧縮オブジェクトのキャッシュの設定が有効になっているため、正規化された `Accept-Encoding` ヘッダーも含まれます。詳細については、「圧縮のサポート」を参照してください。
- キャッシュキーに含まれる Cookie: すべての Cookie が含まれます。
- キャッシュキーに含まれるクエリ文字列: すべてのクエリ文字列が含まれます。
- 圧縮オブジェクトのキャッシュの設定: 有効。詳細については、「圧縮のサポート」を参照してください。

キャッシュキーを理解する

キャッシュキーは、CloudFront エッジロケーションへのビューワーリクエストによってキャッシュヒットが発生するかどうかを決定します。キャッシュキーは、キャッシュ内のオブジェクトの固有の識別子です。キャッシュ内の各オブジェクトには、固有のキャッシュキーがあります。

キャッシュヒットは、ビューワーリクエストが以前のリクエストと同じキャッシュキーを生成し、そのキャッシュキーのオブジェクトがエッジロケーションのキャッシュにあり、有効な場合に発生します。キャッシュヒットが発生すると、リクエストされたオブジェクトが CloudFront エッジロケーションからビューワーに提供されます。これには以下のような利点があります。

- オリジンサーバーの負荷を軽減

- ビューワーのレイテンシーを低減

キャッシュヒット率が高い (キャッシュヒットにつながるビューワーリクエストの割合が高い) ほど、ウェブサイトやアプリケーションのパフォーマンスが高くなります。キャッシュヒット率を改善する 1 つの方法は、キャッシュキーに必要な最小値のみを含めることです。詳細については、次のセクションを参照してください。

キャッシュポリシーを使用して、キャッシュキーの値 (URL クエリ文字列、HTTP ヘッダー、および Cookie) を変更できます。(Lambda@Edge 関数を使用してキャッシュキーを変更することもできます)。キャッシュキーを変更する前に、アプリケーションの設計方法と、ビューワーリクエストの特性に基づいてさまざまなレスポンスをいつ、どのように処理するかを理解することが重要です。ビューワーリクエストの値によってオリジンが返すレスポンスが決定された場合は、その値をキャッシュキーに含める必要があります。しかし、オリジンが返すレスポンスに影響を与えない値をキャッシュキーに含めると、重複したオブジェクトをキャッシュしてしまう可能性があります。

デフォルトのキャッシュキー

デフォルトでは、CloudFront ディストリビューションのキャッシュキーには以下の情報が含まれます。

- CloudFront ディストリビューションのドメイン名 (d111111abcdef8.cloudfront.net など)
- リクエストされたオブジェクトの URL パス (例: `/content/stories/example-story.html`)

> **Note**
>
> この OPTIONS メソッドは、OPTIONS リクエストのキャッシュキーに含まれます。つまり、OPTIONS リクエストへのレスポンスは、GET リクエストおよび HEAD リクエストへのレスポンスとは別にキャッシュされます。

デフォルトでは、ビューワーリクエストのその他の値はキャッシュキーに含まれません。ウェブブラウザからの次の HTTP リクエストを考えてみましょう。

```
          GET /content/stories/example-story.html?ref=0123abc&split-pages=false
HTTP/1.1
          Host: d111111abcdef8.cloudfront.net
          User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/68.0
```

```
            Accept: text/html,*/*
            Accept-Language: en-US,en
            Cookie: session_id=01234abcd
            Referer: https://news.example.com/
```

この例のようなビューワーリクエストが CloudFront エッジロケーションに到達すると、CloudFront はキャッシュキーを使用してキャッシュヒットがあるかどうかを判断します。デフォルトでは、リクエストの `/content/stories/example-story.html` および `d111111abcdef8.cloudfront.net` コンポーネントのみがキャッシュキーに含まれます。リクエストされたオブジェクトがキャッシュにない場合 (キャッシュミス)、CloudFront は、オブジェクトを取得するためのリクエストをオリジンに送信します。オブジェクトを取得した後、CloudFront はビューワーに返し、エッジロケーションのキャッシュに保存します。

CloudFront が、キャッシュキーによって決定された同じオブジェクトに対する別のリクエストを受信すると、CloudFront はオリジンにリクエストを送信することなく、キャッシュされたオブジェクトをビューワーに即座に配信します。たとえば、前のリクエストの後に入ってくる次の HTTP リクエストを考えてみましょう。

```
            GET /content/stories/example-story.html?ref=xyz987&split-pages=true
HTTP/1.1
            Host: d111111abcdef8.cloudfront.net
            User-Agent: Mozilla/5.0 AppleWebKit/537.36 Chrome/83.0.4103.116
            Accept: text/html,*/*
            Accept-Language: en-US,en
            Cookie: session_id=wxyz9876
            Referer: https://rss.news.example.net/
```

このリクエストは、前のリクエストと同じオブジェクトに対するものですが、前のリクエストとは異なります。これは、異なる URL クエリ文字列、異なる `Referer` ヘッダーと `User-Agent` ヘッダー、および異なる `session_id` Cookie を持っています。ただし、これらの値はデフォルトではキャッシュキーの一部ではないので、この 2 番目のリクエストはキャッシュヒットになります。

キャッシュキーをカスタマイズする

場合によっては、キャッシュキーにさらに多くの情報を含める必要がありますが、そうするとキャッシュのヒット数が少なくなることがあります。キャッシュポリシーを使用して、キャッシュキーに含める内容を指定します。

たとえば、オリジンサーバーがビューワーリクエストで Accept-Language HTTP ヘッダーを使用して、ビューワーの言語に基づいて異なるコンテンツを返す場合、このヘッダーをキャッシュキーに含めることができます。これを行うと、CloudFront はこのヘッダーを使用してキャッシュヒットを特定し、オリジンリクエスト (キャッシュミスが発生したときにオリジン CloudFront に送信するリクエスト) にヘッダーを含めます。

キャッシュキーに追加の値を含めると、ビューワーリクエストで発生する可能性のある変化により、CloudFront が重複したオブジェクトをキャッシュする可能性があります。たとえば、ビューワーは、Accept-Language ヘッダーに対して次の値のいずれかを送信できます。

- en-US,en
- en,en-US
- en-US, en
- en-US

これらの異なる値はすべて、ビューワーの言語が英語であることを示していますが、変化によって CloudFront が同じオブジェクトを複数回キャッシュする可能性があります。これにより、キャッシュヒットを減らし、オリジンリクエストの数を増やすことができます。キャッシュキーに Accept-Language ヘッダーを含めず、異なる言語のコンテンツに異なる URL を使用するようにウェブサイトまたはアプリケーションを設定することで、この重複を避けることができます (例: /en-US/content/stories/example-story.html)。

キャッシュキーに含める任意の値については、その値のバリエーションがビューワーリクエストに表示される可能性があることを理解しておく必要があります。特定のリクエスト値については、キャッシュキーに含めることはほとんど意味がありません。たとえば、User-Agent ヘッダーには何千もの固有のバリエーションがあるため、通常はキャッシュキーに含めるには適していません。ユーザー固有の値またはセッション固有の値を持ち、何千もの (または数百万) のリクエストで固有の Cookie も、キャッシュキーを含む候補には適していません。キャッシュキーにこれらの値を含めると、それぞれ固有のバリエーションによって、キャッシュ内のオブジェクトの別のコピーが作成されます。オブジェクトのこれらのコピーが固有でない場合、またはわずかに異なるオブジェクトが多数作成され、各オブジェクトが少数のキャッシュヒットのみを取得する場合は、別のアプローチを検討することをお勧めします。これらの非常に可変性の高い値をキャッシュキーから除外することも、オブジェクトをキャッシュ不可能としてマークすることもできます。

キャッシュキーをカスタマイズするときは注意が必要です。時には望ましいことがありますが、重複オブジェクトのキャッシュ、キャッシュヒット率の低下、オリジンリクエスト数の増加など、意図しない結果が生じる可能性があります。オリジンのウェブサイトまたはアプリケーションが、分析、テ

レメトリー、またはその他の用途に対するビューワーリクエストから特定の値を受け取る必要があるが、これらの値によってオリジンが返すオブジェクトが変更されない場合は、オリジンリクエストポリシーを使用してこれらの値をオリジンリクエストに含めますが、キャッシュキーには含めません。

ポリシーを使用してオリジンリクエストを制御する

CloudFront へのビューワーリクエストにより、キャッシュミス (リクエストされたオブジェクトがエッジロケーションでキャッシュされない) が発生した場合、CloudFront はオブジェクトを取得するためのリクエストをオリジンに送信します。これは、オリジンリクエストと呼ばれます。オリジンリクエストには、ビューワークエストの次の情報が常に含まれます。

- URL パス (URL クエリ文字列またはドメイン名を含まないパスのみ)
- リクエストボディ (存在する場合)
- CloudFront がすべてのオリジンリクエストに自動的に含める HTTP ヘッダー (`Host`、`User-Agent`、`X-Amz-Cf-Id` など)

ビューワーリクエストのその他の情報 (URL クエリ文字列、HTTP ヘッダー、Cookie など) は、デフォルトではオリジンリクエストに含まれません。(例外:レガシーキャッシュ設定では、CloudFront はデフォルトでヘッダーをオリジンに転送します。) ただし、分析やテレメトリーのためにデータを収集するなど、オリジンでこのその他の情報を受信することができます。オリジンリクエストポリシー を使用して、オリジンリクエストに含まれる情報を制御できます。

オリジンリクエストポリシーは、キャッシュキーを制御するキャッシュポリシーとは別のものです。これにより、オリジンで追加情報を受信できます。さらに適切なキャッシュヒット率 (キャッシュヒットとなるビューワーリクエストの割合) を維持できます。これを行うには、オリジンリクエストに含める情報 (オリジンリクエストポリシーを使用) とキャッシュキーに含める情報 (キャッシュポリシーを使用) を個別に制御します。

2 種類のポリシーは別々のものですが、関連性があります。キャッシュキーに含めるすべての URL クエリ文字列、HTTP ヘッダー、および Cookie (キャッシュポリシーを使用) は、オリジンリクエストに自動的に含まれます。オリジンリクエストポリシーを使用して、オリジンリクエストに含めるが、キャッシュキーには含めない情報を指定します。キャッシュポリシーと同様に、オリジンリクエストポリシーを CloudFront ディストリビューション内の 1 つ以上のキャッシュ動作にアタッチします。

オリジンリクエストポリシーを使用して、ビューワーリクエストに含まれていないオリジンリクエストに追加の HTTP ヘッダーを追加することもできます。これらの追加ヘッダーは、オリジンリクエストを送信する前に CloudFront によって追加されます。ヘッダー値は、ビューワーリクエストに基づいて自動的に決定されます。詳細については、「the section called “CloudFront のリクエストヘッダーを追加する”」を参照してください。

トピック

オリジンリクエストポリシーを理解する

CloudFront には、一般的ユースケース用にマネージドポリシーと呼ばれる事前定義されたオリジンリクエストポリシーがいくつか用意されています。これらのマネージドポリシーを使用することも、ユーザーのニーズ別に独自のオリジンリクエストポリシーを作成することもできます。マネージドポリシーの詳細については、「マネージドオリジンリクエストポリシーを使用する」を参照してください。

オリジンリクエストポリシーには以下の設定が含まれます。設定は、ポリシー情報と オリジンリクエスト設定に分類されます。

ポリシー情報

名前

オリジンリクエストポリシーを識別する名前。コンソールでは、名前を使用して、オリジンリクエストポリシーをキャッシュ動作にアタッチします。

説明

オリジンリクエストポリシーを説明するコメント。これはオプションです。

オリジンリクエスト設定

オリジンリクエスト設定では、CloudFront がオリジンに送信するリクエスト (オリジンリクエストと呼ばれる) に含まれるビューワーリクエストの値を指定します。値には、URL クエリ文字列、HTTP ヘッダー、および Cookie を含めることができます。指定した値は、オリジンリクエストに含まれますが、キャッシュキーには含まれません。キャッシュキーの制御については、「ポリシーを使用してキャッシュキーを制御する」を参照してください。

ヘッダー

CloudFront によりオリジンリクエストに含まれる、ビューワーリクエストの HTTP ヘッダー。ヘッダーには、以下のいずれかの設定を選択できます。

- [Note (なし)] – ビューワーリクエストの HTTP ヘッダーは、オリジンリクエストに含まれません。
- [All viewer headers (すべてのビューワーヘッダー)] – ビューワーリクエストのすべての HTTP ヘッダーは、オリジンリクエストに含まれます。
- [すべてのビューワーヘッダーと次の CloudFront ヘッダー] – ビューワーリクエストのすべての HTTP ヘッダーが、オリジンリクエストに含まれます。さらに、オリジンリクエストに追加する CloudFront ヘッダーを指定します。CloudFront ヘッダーの詳細については、「the section called “CloudFront のリクエストヘッダーを追加する”」を参照してください。
- [次のヘッダーを含める] – オリジンリクエストに含める HTTP ヘッダーを指定します。

> **Note**
>
> [オリジンのカスタムヘッダー] 設定に既に含まれているヘッダーを指定しないでください。詳細については、「オリジンリクエストにカスタムヘッダーを追加するように CloudFront を設定する」を参照してください。

- [以下を除くすべてのビューワーヘッダー] — オリジンリクエストに含まれない HTTP ヘッダーを指定します。指定されたものを除いて、ビューワーリクエストの他のすべての HTTP ヘッダーが含まれます。

[すべてのビューワーヘッダーと次の CloudFront ヘッダー]、[次のヘッダーを含める] または [以下を除くすべてのビューワーヘッダー] の設定を使用する場合、HTTP ヘッダーはヘッダー名のみで指定します。CloudFront では、オリジンリクエストに、その値を含む完全なヘッダーが含まれます。

> **Note**
>
> [以下を除くすべてのビューワーヘッダー] 設定を使用してビューワーの `Host` ヘッダーを削除すると、CloudFront はオリジンのドメイン名を含む新しい `Host` ヘッダーをオリジンリクエストに追加します。

Cookie

CloudFront によりオリジンリクエストに含まれる、ビューワーリクエストの Cookie。Cookie には、以下のいずれかの設定を選択できます。

- [None (なし)] – ビューワーリクエストの Cookie は、オリジンリクエストに含まれません。
- [All (すべて)] – ビューワーリクエストのすべての Cookie は、オリジンリクエストに含まれます。
- [次のヘッダーを含める] – ビューワーリクエストのどの Cookie をオリジンリクエストに含めるかを指定します。
- [次を除くすべての Cookie] – ビューワーリクエストのどの Cookie をオリジンリクエストに含めないかを指定します。ビューワーリクエストの他のすべての Cookie が含まれます。

[次のヘッダーを含める] または [次を除くすべての Cookie] 設定を使用する場合、Cookie の名前のみで指定します。CloudFront では、オリジンリクエストに、その値を含む完全な Cookie が含まれます。

クエリ文字列

CloudFront によりオリジンリクエストに含まれる、ビューワーリクエストの URL クエリ文字列。クエリ文字列には、以下のいずれかの設定を選択できます。

- [None (なし)] – ビューワーリクエストのクエリ文字列は、オリジンリクエストに含まれません。
- [All (すべて)] – ビューワーリクエストのすべてのクエリ文字列は、オリジンリクエストに含まれます。
- [次のクエリ文字列を含める] - ビューワーリクエストのどのクエリ文字列をオリジンリクエストに含めるかを指定します。
- [次を除くすべてのクエリ文字列] - ビューワーリクエストのどのクエリ文字列をオリジンリクエストに含めないかを指定します。その他すべてのクエリ文字列が含まれます。

[次のクエリ文字列を含める] または [次を除くすべてのクエリ文字列] 設定を使用する場合、クエリ文字列の名前のみで指定します。CloudFront では、オリジンリクエストに、その値を含む完全なクエリ文字列が含まれます。

オリジンリクエストポリシーを作成する

オリジンリクエストポリシーを使用して、CloudFront がオリジンに送信するリクエストに含まれる値 (URL クエリ文字列、HTTP ヘッダー、Cookie) を制御できます。オリジンリクエストポリシー

は、CloudFront コンソール、AWS Command Line Interface (AWS CLI)、または CloudFront API を使用して作成できます。

オリジンリクエストポリシーを作成したら、CloudFront ディストリビューションの 1 つ以上のキャッシュ動作にアタッチします。

オリジンリクエストポリシーは必須ではありません。キャッシュ動作にオリジンリクエストポリシーがアタッチされていない場合、オリジンリクエストには、キャッシュポリシーで指定されたすべての値が含まれますが、それ以上は含まれません。

> **Note**
>
> オリジンリクエストポリシーを使用するには、キャッシュ動作でもキャッシュポリシーを使用する必要があります。キャッシュポリシーがないと、キャッシュ動作でオリジンリクエストポリシーを使用することはできません。

Console

オリジンリクエストポリシーを作成するには (コンソール)

1. AWS Management Console にログインし、https://console.aws.amazon.com/cloudfront/v4/home?#/policies で CloudFront コンソールの [Policies] (ポリシー) ページを開きます。
2. [オリジンリクエスト] を選択してから、[オリジンリクエストポリシーの作成] を選択します。
3. このオリジンリクエストポリシーに目的の設定を選択します。詳細については、「オリジンリクエストポリシーを理解する」を参照してください。
4. 終了したら、[作成] を選択します。

オリジンリクエストポリシーを作成したら、それをキャッシュ動作にアタッチできます。

オリジンリクエストポリシーを既存のディストリビューションにアタッチするには (コンソール)

1. https://console.aws.amazon.com/cloudfront/v4/home#/distributions で CloudFront コンソールの [Distributions] (ディストリビューション) ページを開きます。
2. 更新するディストリビューションを選択し、[動作] タブを選択します。
3. 更新するキャッシュ動作を選択し、[編集] を選択します。

または、新しいキャッシュ動作を作成するには、[動作を作成] を選択します。

4. [キャッシュキーとオリジンリクエスト] セクションで、[キャッシュポリシーとオリジンリクエストポリシー] が選択されていることを確認します。
5. [オリジンリクエストポリシー] では、このキャッシュ動作にアタッチするオリジンリクエストポリシーを選択します。
6. ページの最下部で [変更の保存] を選択します。

オリジンリクエストポリシーを新しいディストリビューションにアタッチするには (コンソール)

1. で CloudFront コンソールを開きますhttps://console.aws.amazon.com/cloudfront/v4/home
2. [ディストリビューションの作成] を選択します。
3. [キャッシュキーとオリジンリクエスト] セクションで、[キャッシュポリシーとオリジンリクエストポリシー] が選択されていることを確認します。
4. [Origin request policy] (オリジンリクエストポリシー) で、このディストリビューションのデフォルトのキャッシュ動作にアタッチするオリジンリクエストポリシーを選択します。
5. オリジン、デフォルトのキャッシュ動作、その他のディストリビューション設定に必要な設定を選択します。詳細については、「ディストリビューション設定リファレンス」を参照してください。
6. 終了したら、[ディストリビューションの作成] を選択します。

CLI

AWS Command Line Interface (AWS CLI) でオリジンリクエストポリシーを作成するには、aws cloudfront create-origin-request-policy コマンドを使用します。コマンドの入力パラメータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

オリジンリクエストポリシーを作成するには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、`origin-request-policy.yaml` コマンドのすべての入力パラメータを含む create-origin-request-policy という名前のファイルを作成します。

```
aws cloudfront create-origin-request-policy --generate-cli-skeleton yaml-input >
 origin-request-policy.yaml
```

2. 先ほど作成した origin-request-policy.yaml という名前のファイルを開きます。ファイルを編集して、必要なオリジンリクエストポリシー設定を指定し、ファイルを保存します。ファイルからオプションのフィールドを削除することはできますが、必須フィールドは削除しないでください。

   オリジンリクエストポリシー設定の詳細については、「オリジンリクエストポリシーを理解する」を参照してください。

3. 次のコマンドを使用して、origin-request-policy.yaml ファイルの入力パラメータを使用し、オリジンリクエストポリシーを作成します。

   ```
   aws cloudfront create-origin-request-policy --cli-input-yaml file://origin-
   request-policy.yaml
   ```

   コマンドの出力の Id 値を書き留めます。これはオリジンリクエストポリシー ID であり、オリジンリクエストポリシーを CloudFront ディストリビューションのキャッシュ動作にアタッチするために必要です。

オリジンリクエストポリシーを既存のディストリビューションにアタッチするには (入力ファイルを含む CLI)

1. 以下のコマンドを使用して、更新する CloudFront ディストリビューションのディストリビューション設定を保存します。*distribution_ID* をディストリビューションの ID に置き換えます。

   ```
   aws cloudfront get-distribution-config --id distribution_ID --output yaml >
    dist-config.yaml
   ```

2. 先ほど作成した dist-config.yaml という名前のファイルを開きます。ファイルを編集し、オリジンリクエストポリシーを使用するように更新する各キャッシュ動作に次の変更を加えます。

   - キャッシュ動作で、OriginRequestPolicyId という名前のフィールドを追加します。フィールドの値には、ポリシーの作成後に書き留めたオリジンリクエストポリシー ID を使用します。
   - ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. オリジンリクエストポリシーを使用するようにディストリビューションを更新するには、次のコマンドを使用します。*distribution_ID* をディストリビューションの ID に置き換えます。

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

オリジンリクエストポリシーを新しいディストリビューションにアタッチするには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、distribution.yaml コマンドのすべての入力パラメータを含む create-distribution という名前のファイルを作成します。

```
aws cloudfront create-distribution --generate-cli-skeleton yaml-input >
 distribution.yaml
```

2. 先ほど作成した distribution.yaml という名前のファイルを開きます。デフォルトのキャッシュ動作の [OriginRequestPolicyId] フィールドに、ポリシーの作成後に書き留めたオリジンリクエストポリシー ID を入力します。ファイルの編集を続行して必要なディストリビューション設定を指定し、完了したらファイルを保存します。

   ディストリビューション設定の詳細については、「ディストリビューション設定リファレンス」を参照してください。

3. 次のコマンドを使用して、distribution.yaml ファイルの入力パラメータを使用し、ディストリビューションを作成します。

```
aws cloudfront create-distribution --cli-input-yaml file://distribution.yaml
```

API

CloudFront API を使用してオリジンリクエストポリシーを作成するには、CreateOriginRequestPolicy を使用します。この API コールで指定するフィールドの詳細につ

いては、「オリジンリクエストポリシーを理解する」、および AWS SDK やその他の API クライアントの API リファレンスドキュメントを参照してください。

オリジンリクエストポリシーを作成したら、次の API コールのいずれかを使用して、それをキャッシュ動作にアタッチできます。

- 既存のディストリビューションのキャッシュ動作にアタッチするには、UpdateDistribution を使用します。
- 新しいディストリビューションのキャッシュ動作にアタッチするには、CreateDistribution を使用します。

これらの API コールの両方について、キャッシュ動作内で、`OriginRequestPolicyId` フィールドにオリジンリクエストポリシーの ID を指定します。これらの API コールで指定するその他フィールドの詳細については、「ディストリビューション設定リファレンス」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

マネージドオリジンリクエストポリシーを使用する

CloudFront は、ディストリビューションのキャッシュ動作にアタッチできる一連のマネージドオリジンリクエストポリシーを用意しています。マネージドオリジンリクエストポリシーを使用すると、独自のオリジンリクエストポリシーを記述したり、維持したりする必要はありません。マネージドポリシーでは、特定のユースケースに最適化された設定を使用します。

マネージドオリジンリクエストポリシーを使用するには、ディストリビューションのキャッシュ動作にそのポリシーをアタッチします。このプロセスは、オリジンリクエストポリシーを作成するときと同じですが、新しいオリジンリクエストポリシーを作成するのではなく、マネージドオリジンリクエストポリシーの 1 つをアタッチするだけです。ポリシーをアタッチするには、名前 (コンソールの場合) または ID (AWS CLI または SDK の場合) を使用します。名前と ID は、次のセクションに記載されています。

詳細については、「オリジンリクエストポリシーを作成する」を参照してください。

次のトピックでは、使用可能なマネージドオリジンリクエストポリシーについて説明します。

トピック

- AllViewer
- AllViewerAndCloudFrontHeaders-2022-06

- AllViewerExceptHostHeader
- CORS-CustomOrigin
- CORS-S3Origin
- Elemental-MediaTailor-PersonalizedManifests
- UserAgentRefererHeaders

AllViewer

このポリシーを CloudFront コンソールで見る

このポリシーには、ビューワーリクエストのすべての値 (ヘッダー、Cookie、クエリ文字列) が含まれます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
216adef6-5c7f-47e4-b989-5492eafa07d3
```

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー: ビューワーリクエスト内のすべてのヘッダー
- オリジンリクエストに含まれる Cookie: すべて
- オリジンリクエストに含まれるクエリ文字列: すべて

AllViewerAndCloudFrontHeaders-2022-06

このポリシーを CloudFront コンソールで見る

このポリシーには、ビューワーリクエストのすべての値 (ヘッダー、Cookie、クエリ文字列) と、2022 年 6 月までにリリースされたすべての CloudFront ヘッダーが含まれます (2022 年 6 月の後にリリースされた CloudFront ヘッダーは含まれません)。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
33f36d7e-f396-46d9-90e0-52428a34d9dc
```

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー: ビューワーリクエスト内のすべてのヘッダー、および以下の CloudFront ヘッダー:
  - `CloudFront-Forwarded-Proto`
  - `CloudFront-Is-Android-Viewer`
  - `CloudFront-Is-Desktop-Viewer`
  - `CloudFront-Is-IOS-Viewer`
  - `CloudFront-Is-Mobile-Viewer`
  - `CloudFront-Is-SmartTV-Viewer`
  - `CloudFront-Is-Tablet-Viewer`
  - `CloudFront-Viewer-Address`
  - `CloudFront-Viewer-ASN`
  - `CloudFront-Viewer-City`
  - `CloudFront-Viewer-Country`
  - `CloudFront-Viewer-Country-Name`
  - `CloudFront-Viewer-Country-Region`
  - `CloudFront-Viewer-Country-Region-Name`
  - `CloudFront-Viewer-Http-Version`
  - `CloudFront-Viewer-Latitude`
  - `CloudFront-Viewer-Longitude`
  - `CloudFront-Viewer-Metro-Code`
  - `CloudFront-Viewer-Postal-Code`
  - `CloudFront-Viewer-Time-Zone`
  - `CloudFront-Viewer-TLS`
- オリジンリクエストに含まれる Cookie: すべて
- オリジンリクエストに含まれるクエリ文字列: すべて

AllViewerExceptHostHeader

このポリシーを CloudFront コンソールで見る

このポリシーには、ビューワーリクエストの `Host` ヘッダーは含まれませんが、ビューワーリクエストの他のすべての値 (ヘッダー、Cookie、クエリ文字列) が含まれます。

このポリシーには、HTTP プロトコル、HTTP バージョン、TLS バージョン用の追加の CloudFront リクエストヘッダーと、デバイスタイプやビューワーの場所を特定するためのすべてのヘッダーも含まれます。

このポリシーは、Amazon API Gateway と AWS Lambda 関数 URL オリジンでの使用を目的としています。これらのオリジンは、Host ヘッダーに CloudFront ディストリビューションのドメイン名ではなく、オリジンドメイン名が含まれていることを前提としています。ビューワーリクエストの Host ヘッダーをこれらのオリジンに転送すると、それらが機能しなくなる可能性があります。

> **Note**
>
> このマネージドオリジンリクエストポリシーを使用してビューワーの Host ヘッダーを削除すると、CloudFront はオリジンのドメイン名を含む新しい Host ヘッダーをオリジンリクエストに追加します。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
b689b0a8-53d0-40ab-baf2-68738e2966ac
```

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー: Host ヘッダーを除くビューワーリクエスト内のすべてのヘッダー
- オリジンリクエストに含まれる Cookie: すべて
- オリジンリクエストに含まれるクエリ文字列: すべて

CORS-CustomOrigin

このポリシーを CloudFront コンソールで見る

このポリシーには、オリジンがカスタムオリジンである場合に、Cross-Origin Resource Sharing (CORS) リクエストを有効にするヘッダーが含まれます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
59781a5b-3903-41f3-afcb-af62929ccde1
```

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー:
  - `Origin`
- オリジンリクエストに含まれる Cookie: なし
- オリジンリクエストに含まれるクエリ文字列: なし

CORS-S3Origin

このポリシーを CloudFront コンソールで見る

このポリシーには、オリジンが Amazon S3 バケットである場合に、Cross-Origin Resource Sharing (CORS) リクエストを有効にするヘッダーが含まれます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

`88a5eaf4-2fd4-4709-b370-b4c650ea3fcf`

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー:
  - `Origin`
  - `Access-Control-Request-Headers`
  - `Access-Control-Request-Method`
- オリジンリクエストに含まれる Cookie: なし
- オリジンリクエストに含まれるクエリ文字列: なし

Elemental-MediaTailor-PersonalizedManifests

このポリシーを CloudFront コンソールで見る

このポリシーは、オリジンとしての AWS Elemental MediaTailor エンドポイント用に意図されています。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

775133bc-15f2-49f9-abea-afb2e0bf67d2

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー:
  - Origin
  - Access-Control-Request-Headers
  - Access-Control-Request-Method
  - User-Agent
  - X-Forwarded-For
- オリジンリクエストに含まれる Cookie: なし
- オリジンリクエストに含まれるクエリ文字列: すべて

UserAgentRefererHeaders

このポリシーを CloudFront コンソールで見る

このポリシーには、User-Agent ヘッダーと Referer ヘッダーのみが含まれます。クエリ文字列や Cookie は含まれません。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

acba4595-bd28-49b8-b9fe-13317c0390fa

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー:
  - User-Agent
  - Referer
- オリジンリクエストに含まれる Cookie: なし
- オリジンリクエストに含まれるクエリ文字列: なし

CloudFront のリクエストヘッダーを追加する

CloudFront がビューワーから受け取り、オリジンまたはエッジ関数に転送するリクエストに特定の HTTP ヘッダーを追加するように CloudFront を設定できます。これらの HTTP ヘッダーの値は、

ビューワーまたはビューワーリクエストの特性に基づいています。ヘッダーは、ビューワーのデバイスタイプ、IP アドレス、地理的位置、リクエストプロトコル (HTTP または HTTPS)、HTTP バージョン、TLS 接続の詳細、および JA3 フィンガープリントに関する情報を提供します。

これらのヘッダーを使用すると、オリジンまたはエッジ関数でビューワーに関する情報を受け取ることができ、この情報を特定するための独自のコードを記述する必要はありません。オリジンが、これらのヘッダーの情報に基づいてレスポンスを返す場合は、CloudFront が別のレスポンスを別々にキャッシュするように、それらをキャッシュキーに含めることができます。例えば、オリジンは、ビューワーが居住する国に基づく特定の言語のコンテンツや、特定のデバイスタイプに合わせたコンテンツで応答する場合があります。オリジンは、これらのヘッダーをログファイルに書き込むこともあります。ログファイルを使用して、ビューワーの場所、ビューワーのデバイスの種類などの情報を判断できます。

キャッシュキーにこれらのヘッダーを含めるには、キャッシュポリシーを使用します。詳細については、ポリシーを使用してキャッシュキーを制御するおよびthe section called “キャッシュキーを理解する”を参照してください。

オリジンでヘッダーを受信するが、キャッシュキーに含めないようにするには、オリジンリクエストポリシーを使用します。詳しくは、「ポリシーを使用してオリジンリクエストを制御する」を参照してください。

トピック

- ビューワーのデバイスタイプを特定するためのヘッダー
- ビューワーの場所を特定するためのヘッダー
- ビューワーのヘッダー構造を特定するためのヘッダー
- その他の CloudFront ヘッダー

ビューワーのデバイスタイプを特定するためのヘッダー

ビューワーのデバイスタイプを特定するために、次のヘッダーが追加できます。`User-Agent` ヘッダーの値に基づいて、CloudFront はこれらのヘッダーの値を `true` または `false` に設定します。デバイスが複数のカテゴリに属していると、複数の値が `true` になる場合があります。例えば、一部のタブレットデバイスに対して、CloudFront が `CloudFront-Is-Mobile-Viewer` と `CloudFront-Is-Tablet-Viewer` の両方を `true` に設定します。

- `CloudFront-Is-Android-Viewer` - ビューワーが Android オペレーティングシステム搭載のデバイスであると CloudFront が判断すると、`true` に設定します。

- `CloudFront-Is-Desktop-Viewer` - ビューワーがデスクトップデバイスであると CloudFront が判断すると、`true` に設定します。
- `CloudFront-Is-IOS-Viewer` – ビューワーが iPhone、iPod touch、その他の iPad デバイスなど、Apple モバイルオペレーティングシステム搭載のデバイスであると CloudFront が判断すると、`true` に設定します。
- `CloudFront-Is-Mobile-Viewer` - ビューワーがモバイルデバイスであると CloudFront が判断すると、`true` に設定します。
- `CloudFront-Is-SmartTV-Viewer` - ビューワーがスマートテレビであると CloudFront が判断すると、`true` に設定します。
- `CloudFront-Is-Tablet-Viewer` - ビューワーがタブレットであると CloudFront が判断すると、`true` に設定します。

ビューワーの場所を特定するためのヘッダー

ビューワーの場所を特定するために、以下のヘッダーが追加できます。CloudFront は、ビューワーの IP アドレスに基づいてこれらのヘッダーの値を決定します。これらのヘッダーの値が ASCII 以外の文字の場合、CloudFront パーセントは RFC 3986 のセクション 1.2 に従って文字をパーセントエンコードします。

- `CloudFront-Viewer-Address` – ビューワーの IP アドレスと、リクエストのソースポートを示します。例えば、`198.51.100.10:46532` のヘッダー値は、ビューワーの IP アドレスが 198.51.100.10 で、リクエストのソースポートが 46532 であることを意味します。
- `CloudFront-Viewer-ASN` — ビューワーの AS 番号 (ASN) を示します。

> **Note**
>
> `CloudFront-Viewer-Address` と `CloudFront-Viewer-ASN` は、オリジンリクエストポリシーで追加できますが、キャッシュポリシーでは追加できません。

- `CloudFront-Viewer-Country` - ビューワーの国の 2 文字の国コードが含まれています。国コードの一覧については、「ISO 3166-1 alpha-2」を参照してください。
- `CloudFront-Viewer-City` - ビューワーの市区町村名が含まれています。

以下のヘッダーを追加すると、CloudFrontは、AWS ネットワークから発信されたリクエストを除くすべてのリクエストにそれらを適用します。

- CloudFront-Viewer-Country-Name - ビューワーの国名が含まれています。
- CloudFront-Viewer-Country-Region - ビューワーのリージョンを表すコード (最大 3 文字) が含まれています。地域は、ISO 3166-2コードの第 1 レベルのサブディビジョン (最も広いまたは狭い) です。
- CloudFront-Viewer-Country-Region-Name - ビューワーのリージョン名が含まれています。地域は、ISO 3166-2コードの第 1 レベルのサブディビジョン (最も広いまたは狭い) です。
- CloudFront-Viewer-Latitude - ビューワーのおよその緯度が含まれています。
- CloudFront-Viewer-Longitude - ビューワーのおよその経度が含まれています。
- CloudFront-Viewer-Metro-Code - ビューワーのメトロコードが含まれています。これは、ビューワーが米国にいる場合にのみ表示されます。
- CloudFront-Viewer-Postal-Code - ビューワーの郵便番号が含まれています。
- CloudFront-Viewer-Time-Zone IANA タイムゾーンデータベース形式 (例: America/Los_Angeles) で、ビューワーのタイムゾーンが含まれています。

ビューワーのヘッダー構造を特定するためのヘッダー

送信するヘッダーに基づいてビューワーを特定しやすくするために、以下のヘッダーを追加できます。例えば、異なるブラウザごとに HTTP ヘッダーを送信する順序は異なる場合があります。User-Agent ヘッダーに指定されているブラウザが、そのブラウザの想定されたヘッダーの順序と一致しない場合は、リクエストを拒否できます。また、CloudFront-Viewer-Header-Count 値が CloudFront-Viewer-Header-Order 内のヘッダー数と一致しない場合は、リクエストを拒否できます。

- CloudFront-Viewer-Header-Order — ビューワーのヘッダー名をリクエスト順にコロンで区切って示します。例: CloudFront-Viewer-Header-Order: Host:User-Agent:Accept:Accept-Encoding。文字数の制限である 7,680 文字を超えるヘッダーは切り捨てられます。
- CloudFront-Viewer-Header-Count — ビューワーのヘッダーの総数を示します。

その他の CloudFront ヘッダー

ビューワーのプロトコル、バージョン、JA3 フィンガープリント、および TLS 接続の詳細を特定するために、以下のヘッダーを追加できます。

- `CloudFront-Forwarded-Proto` - ビューワーのリクエストのプロトコル (HTTP または HTTPS) を示します。
- `CloudFront-Viewer-Http-Version` - ビューワーのリクエストの HTTP バージョンを示します。
- `CloudFront-Viewer-JA3-Fingerprint` — ビューワーの JA3 フィンガープリントを示します。JA3 フィンガープリントは、既知のクライアントからのリクエストであるか、マルウェアまたは悪意のあるボットであるか、想定された (許可リストに登録されている) アプリケーションであるかを判断するのに役立ちます。このヘッダーは、ビューワーの SSL/TLS `Client Hello` パケットに依存し、HTTPS リクエスト専用となっています。

  > **Note**
  >
  > `CloudFront-Viewer-JA3-Fingerprint` の追加は、オリジンリクエストポリシーではできますが、キャッシュポリシーではできません。

- `CloudFront-Viewer-TLS` – SSL/TLS バージョン、暗号、およびビューアと CloudFront 間の接続に使用された SSL/TLS ハンドシェイクに関する情報が含まれています。ヘッダー値は次の形式です。

  ```
  SSL/TLS_version:cipher:handshake_information
  ```

  handshake_information の場合、ヘッダーには以下のいずれかの値が含まれます。

  - `fullHandshake` – SSL/TLS セッションに対して完全なハンドシェイクが実行された。
  - `sessionResumed` – 前の SSL/TLS セッションが再開された。
  - `connectionReused` – 前の SSL/TLS 接続が再利用された。

  以下に示しているのは、このヘッダーの値の例です。

  ```
  TLSv1.3:TLS_AES_128_GCM_SHA256:sessionResumed
  ```

  ```
  TLSv1.2:ECDHE-ECDSA-AES128-GCM-SHA256:connectionReused
  ```

  ```
  TLSv1.1:ECDHE-RSA-AES128-SHA256:fullHandshake
  ```

  ```
  TLSv1:ECDHE-RSA-AES256-SHA:fullHandshake
  ```

このヘッダー値に含まれる SSL/TLS バージョンと暗号の完全なリストについては、「the section called “ビューワーと CloudFront との間でサポートされているプロトコルと暗号”」を参照してください。

> **Note**
>
> `CloudFront-Viewer-TLS` の追加は、オリジンリクエストポリシーではできますが、キャッシュポリシーではできません。

オリジンリクエストポリシーとキャッシュポリシーの連携方法を理解する

CloudFront オリジンリクエストポリシーを使用して、CloudFront がオリジンに送信するリクエスト (オリジンリクエストと呼ばれる) を制御できます。オリジンリクエストポリシーを使用するには、同じキャッシュ動作にキャッシュポリシーをアタッチする必要があります。キャッシュポリシーがないと、キャッシュ動作でオリジンリクエストポリシーを使用することはできません。詳細については、「ポリシーを使用してオリジンリクエストを制御する」を参照してください。

オリジンリクエストポリシーとキャッシュポリシーが連携して、CloudFront がオリジンリクエストに含める値を決定します。キャッシュキーに指定するすべての URL クエリ文字列、HTTP ヘッダー、および Cookie (キャッシュポリシーを使用) は、オリジンリクエストに自動的に含まれます。オリジンリクエストポリシーで指定した追加のクエリ文字列、ヘッダー、および Cookie もオリジンリクエストに含まれます (キャッシュキーには含まれません)。

オリジンリクエストポリシーとキャッシュポリシーには、互いに競合しているように見える設定があります。例えば、あるポリシーでは特定の値を許可し、別のポリシーではそれらをブロックするなどです。次の表では、オリジンリクエストポリシーとキャッシュポリシーの設定を一緒に使用する場合に、CloudFront がオリジンリクエストに含める値を説明します。これらの設定は、一般的にすべてのタイプの値 (クエリ文字列、ヘッダー、Cookie) に適用されます。ただし、キャッシュポリシーですべてのヘッダーを指定したり、ヘッダーブロックリストを使用することはできません。

	オリジンリクエストポリシー			
	なし	[All] (すべて)	許可リスト	ブロックリスト
キャッシュポリシー				

	オリジンリクエストポリシー			
	なし	[All] (すべて)	許可リスト	ブロックリスト
なし	すべてのオリジンリクエストに含まれるデフォルトを除き、ビューワーリクエストの値はオリジンリクエストには含まれません。詳細については、「ポリシーを使用してオリジンリクエストを制御する」を参照してください。	ビューワーリクエストのすべての値がオリジンリクエストに含まれます。	オリジンリクエストポリシーで指定された値のみがオリジンリクエストに含まれます。	オリジンリクエストポリシーで指定された値を除くビューワーリクエストのすべての値がオリジンリクエストに含まれます。

	オリジンリクエストポリシー			
	なし	[All] (すべて)	許可リスト	ブロックリスト
[All] (すべて) 注意: キャッシュポリシーですべてのヘッダーを指定することはできません。	ビューワーリクエストのすべてのクエリ文字列と Cookie は、オリジンリクエストに含まれます。	ビューワーリクエストのすべての値がオリジンリクエストに含まれます。	ビューワーリクエストのすべてのクエリ文字列と Cookie、およびオリジンリクエストポリシーで指定されたヘッダーは、オリジンリクエストに含まれます。	ビューワーリクエストのすべてのクエリ文字列と Cookie は、オリジンリクエストポリシーのブロックリストで指定されているものも含め、すべてオリジンリクエストに含まれます。キャッシュポリシーの設定は、オリジンリクエストポリシーのブロックリストより優先されます。

	オリジンリクエストポリシー			
	なし	[All] (すべて)	許可リスト	ブロックリスト
許可リスト	ビューワーリクエストから指定された値のみが、オリジンリクエストに含まれます。	ビューワーリクエストのすべての値がオリジンリクエストに含まれます。	キャッシュポリシーまたはオリジンリクエストポリシーで指定されているすべての値は、オリジンリクエストに含まれます。	キャッシュポリシーで指定された値は、オリジンリクエストポリシーのブロックリストで同じ値が指定されている場合でも、オリジンリクエストに含まれます。キャッシュポリシーの許可リストは、オリジンリクエストポリシーの禁止リストより優先されます。

	オリジンリクエストポリシー			
	なし	[All] (すべて)	許可リスト	ブロックリスト
ブロックリスト 注意: キャッシュポリシーのブロックリストではヘッダーを指定できません。	指定されたものを除くビューワーリクエストのすべてのクエリ文字列と Cookie は、オリジンリクエストに含まれます。	ビューワーリクエストのすべての値がオリジンリクエストに含まれます。	オリジンリクエストポリシーで指定された値は、キャッシュポリシーのブロックリストに同じ値が指定されている場合でも、オリジンリクエストに含まれます。オリジンリクエストポリシーの許可リストは、キャッシュポリシーのブロックリストより優先されます。	キャッシュポリシーまたはオリジンリクエストポリシーで指定された値を除くビューワーリクエストのすべての値がオリジンリクエストに含まれます。

ポリシーを使用して CloudFront レスポンスの HTTP ヘッダーを追加または削除する

ビューワー (ウェブブラウザやその他のクライアント) に送信するレスポンスの HTTP ヘッダーを変更するように CloudFront を設定できます。CloudFront は、レスポンスをビューワーに送信する前に、オリジンから受け取ったヘッダーを削除したり、レスポンスにヘッダーを追加したりできます。これらの変更を行うために、コードを記述したり、オリジンを変更したりする必要はありません。

例えば、`X-Powered-By` や `Vary` などのヘッダーを削除し、これらのヘッダーを含まないレスポンスを CloudFront からビューワーに送信できます。または、以下のような HTTP ヘッダーを追加できます。

- ブラウザのキャッシュを制御する `Cache-Control` ヘッダー。
- Cross-Origin Resource Sharing (CORS) を有効にする `Access-Control-Allow-Origin` ヘッダー また、他の CORS ヘッダーを追加することもできます。
- `Strict-Transport-Security`、`Content-Security-Policy`、`X-Frame-Options` のような一般的なセキュリティヘッダーのセット。
- CloudFront を介したリクエストとレスポンスの両方のパフォーマンスとルーティングに関連する情報を表示する、`Server-Timing` ヘッダー。

CloudFront が HTTP レスポンスで追加または削除するヘッダーを指定するには、レスポンスヘッダーポリシーを使用します。レスポンスヘッダーポリシーをもう 1 つのキャッシュ動作にアタッチすると、CloudFront は、そのキャッシュ動作に一致するリクエストに対して送信するレスポンスのヘッダーを変更します。CloudFront は、キャッシュから提供するレスポンスとオリジンから転送するレスポンスのヘッダーを変更します。レスポンスヘッダーポリシーで追加したヘッダーの 1 つ以上がオリジンレスポンスに含まれている場合、CloudFront がオリジンから受信したヘッダーを使用するか、このヘッダーをレスポンスヘッダーポリシー内のヘッダーで上書きするかをポリシーで指定できます。

CloudFront には、一般的なユースケース用にマネージドポリシーと呼ばれる事前定義されたレスポンスヘッダーポリシーが用意されています。これらのマネージドポリシーを使用することも、独自のポリシーを作成することもできます。AWS アカウント の複数のディストリビューションにおける複数のキャッシュ動作に単一のレスポンスヘッダーポリシーをアタッチすることができます。

詳細については、以下のトピックを参照してください。

トピック

レスポンスヘッダーポリシーを理解する

レスポンスヘッダーポリシーを使用して、Amazon CloudFront からビューワーに送信するレスポンスで削除または追加する HTTP ヘッダーを指定できます。レスポンスヘッダーポリシーおよびそれらを使用する理由の詳細については、「ポリシーを使用してレスポンスヘッダーを追加または削除する」を参照してください。

以下のトピックでは、レスポンスヘッダーポリシーの設定について説明します。設定はカテゴリに分類され、そのことについて次のトピックで説明します。

トピック

ポリシーの詳細 (メタデータ)

ポリシーの詳細設定には、レスポンスヘッダーポリシーに関するメタデータが含まれます。

- 名前 - レスポンスヘッダーポリシーを識別するための名前。コンソールでは、名前を使用して、このポリシーをキャッシュ動作にアタッチします。
- 説明 (オプション) - レスポンスヘッダーポリシーを説明するコメント。これはオプションですが、このポリシーの目的を特定するのに役立ちます。

CORS ヘッダー

Cross-Origin Resource Sharing (CORS) 設定を使用して、レスポンスヘッダーポリシーに CORS ヘッダーを追加および設定できます。

このリストは、レスポンスヘッダーポリシーの設定および有効な値の指定方法に焦点を合わせています。これらの各ヘッダーと、実際の CORS リクエストとレスポンスの使用方法の詳細については、MDN ウェブドキュメントと CORS プロトコルの仕様にある「クロスオリジンリソース共有」を参照してください。

Access-Control-Allow-Credentials

これは Boolean 設定 (`true` または `false`) で、CloudFront が CORS リクエストへのレスポンスに `Access-Control-Allow-Credentials` ヘッダーを追加するかどうかを決定します。この設定が `true` に設定されている場合、CloudFront によって `Access-Control-Allow-Credentials: true` CORS リクエストへのレスポンスのヘッダーが追加されます。そうしないと、このヘッダーは CloudFront によってレスポンスに追加されません。

Access-Control-Allow-Headers

CORS のプリフライトリクエストへのレスポンスで、CloudFront が `Access-Control-Allow-Headers` ヘッダーの値として使用するヘッダー名を指定します。この設定の有効な値には、HTTP ヘッダー名、またはすべてのヘッダーを許可することを示すワイルドカード文字 (*) が含まれます。

> **Note**
>
> `Authorization` ヘッダーはワイルドカードを使用できず、明示的にリストする必要があります。

ワイルドカード文字の有効な使用例

例	一致する	一致しない
`x-amz-*`	`x-amz-test` `x-amz-`	`x-amz`
`x-*-amz`	`x-test-amz`	

例	一致する	一致しない
	x--amz	
*	Authorization を除くすべてのヘッダー	Authorization

Access-Control-Allow-Methods

CORS のプリフライトリクエストへのレスポンスで、CloudFront が Access-Control-Allow-Methods ヘッダーの値として使用する HTTP メソッドを指定します。有効な値は、GET、DELETE、HEAD、OPTIONS、PATCH、POST、PUT、ALL です。ALL は、リストされているすべての HTTP メソッドが含まれている特別な値です。

Access-Control-Allow-Origin

CloudFront が Access-Control-Allow-Origin レスポンスヘッダーで使用できる値を指定します。この設定の有効な値には、特定のオリジン (http://www.example.com など)、またはすべてのオリジンを許可することを示すワイルドカード文字 (*) が含まれます。次の表の例を参照してください。

> **Note**
>
> ワイルドカード文字 (*) は、ドメイン (*.example.org) の左端部分として使用できます。
> ワイルドカード文字 (*) は以下の位置では使用できません。
>
> - 最上位ドメイン (example.*)
> - サブドメイン (test.*.example.org) の右側
> - 規約の内部 (exa*mple.org)

ワイルドカード文字の有効な使用例を次の表に示します。

例	一致する	一致しない
http://*.example.org	http://www.example.org	**https**://test.example.org

例	一致する	一致しない
	http://test.example.org http://test.example.org:123	**https**://test.example.org:123
*.example.org	test.example.org test.test.example.org .example.org http://test.example.org https://test.example.org http://test.example.org:123 https://test.example.org:123	
example.org	http://example.org https://example.org	
http://example.org		**https**://example.org http://example.org:123
http://example.org:*	http://example.org:123 http://example.org	

例	一致する	一致しない

`http://example.org:1*3`	`http://example.org:123`
	
	`http://example.org:1893`
	
	`http://example.org:13`	

`*.example.org:1*`	`test.example.org:123`	

Access-Control-Expose-Headers

CORS のプリフライトリクエストへのレスポンスで、CloudFront が `Access-Control-Expose-Headers` ヘッダーの値として使用するヘッダー名を指定します。この設定の有効な値には、HTTP ヘッダー名、またはワイルドカード文字 (*) が含まれます。

Access-Control-Max-Age

CORS のプリフライトリクエストへのレスポンスで、CloudFront が `Access-Control-Max-Age` ヘッダーの値として使用する秒数。

オリジンのオーバーライド

オリジンからのレスポンスに含まれている CORS ヘッダーの 1 つがポリシー内にもある場合の CloudFront の動作方法を決定する Boolean 設定。

- `true` に設定されていて、オリジンレスポンスで、ポリシーにも CORS ヘッダーが含まれている場合、CloudFront は、レスポンスに対するポリシーの CORS ヘッダーを追加します。次に CloudFront はそのレスポンスをビューワーに送信します。CloudFront はオリジンから受け取ったヘッダーを無視します。
- `false` に設定されており、オリジンレスポンスに CORS ヘッダーが含まれている場合 (CORS ヘッダーがポリシーのものかどうかに関係なく)、CloudFront には、オリジンから受け取りレスポンスに送信する CORS ヘッダーが含まれます。CloudFront は、ビューワーに送信されるレスポンスにポリシー内の CORS ヘッダーを追加しません。

セキュリティヘッダー

セキュリティヘッダーの設定を使用して、レスポンスヘッダーポリシーでいくつかのセキュリティ関連の HTTP レスポンスヘッダーを追加および設定できます。

このリストは、レスポンスヘッダーポリシーの設定および有効な値の指定方法を示しています。これらの各ヘッダー、および実際の HTTP レスポンスでの使用方法の詳細については、MDN ウェブドキュメントへのリンクを参照してください。

Content-Security-Policy

CloudFront が `Content-Security-Policy` レスポンスヘッダーの値として使用するコンテンツセキュリティポリシーディレクティブを指定します。

このヘッダーと有効なポリシーディレクティブの詳細については、MDN Web ドキュメントにある「Content-Security-Policy」を参照してください。

> **Note**
>
> `Content-Security-Policy` ヘッダー値は 1,783 文字に限定されます。

リファラーポリシー

CloudFront が `Referrer-Policy` レスポンスヘッダーの値として使用するリファラーポリシーディレクティブを指定します。この設定の有効値は、`no-referrer`、`no-referrer-when-downgrade`、`origin`、`origin-when-cross-origin`、`same-origin`、`strict-origin`、`strict-origin-when-cross-origin`、`unsafe-url` です。

このヘッダーおよびこれらのディレクティブの詳細については、MDN Web ドキュメントにある「Referrer-Policy」を参照してください。

Strict-Transport-Security

CloudFront が `Strict-Transport-Security` レスポンスヘッダーの値として使用するディレクティブおよび設定を指定します。この設定では、以下を個別に指定します。

- CloudFront がこのヘッダーの `max-age` ディレクティブで値として使用する秒数
- CloudFront がこのヘッダーの値に `preload` ディレクティブを含めるかどうかを決定する `preload` の Boolean の設定 (`true` または `false`)

- CloudFront がこのヘッダーの値に `true` ディレクティブを含めるかどうかを決定する `false` の Boolean の設定 (`includeSubDomains` または `includeSubDomains`)

このヘッダーおよびこれらのディレクティブの詳細については、MDN Web ドキュメントにある「Strict-Transport-Security」を参照してください。

X-Content-Type-Options

これは Boolean 設定 (`true` または `false`) で、CloudFront が CORS リクエストへのレスポンスに `X-Content-Type-Options` ヘッダーを追加するかどうかを決定します。この設定が `true` である場合、CloudFront によってレスポンスに `X-Content-Type-Options: nosniff` ヘッダーが追加されます。そうしないと、このヘッダーは CloudFront によって追加されません。

このヘッダーの詳細については、MDN Web ドキュメントにある「X-Content-Type-Options」を参照してください。

X-Frame-Options

CloudFront が `X-Frame-Options` レスポンスヘッダーの値として使用するディレクティブを指定します。この設定の有効値は、`DENY` または `SAMEORIGIN` です。

このヘッダーおよびこれらのディレクティブの詳細については、MDN Web ドキュメントにある「X-Frame-Options」を参照してください。

X-XSS-Protection

CloudFront が `X-XSS-Protection` レスポンスヘッダーの値として使用するディレクティブおよび設定を指定します。この設定では、以下を個別に指定します。

- `X-XSS-Protection` (XSS フィルタリングを無効にする) または `0` (XSS フィルタリングを有効にする) についての `1` の設定設定
- CloudFront がこのヘッダーの値に `true` ディレクティブを含めるかどうかを決定する `false` の Boolean の設定 (`block` または `mode=block`)
- CloudFront が、このヘッダーの値に `report=`*`reporting URI`* ディレクティブを含めるかどうかを決定するレポートの URI

`block` のために `true` を指定することができます。または、レポートの URI を指定することもできますが、両方一緒に指定することはできません。このヘッダーおよびこれらのディレクティブの詳細については、MDN Web ドキュメントにある「X-XSS-Protection」を参照してください。

オリジンのオーバーライド

これらの各セキュリティヘッダーの設定には、Boolean 設定 (`true` または `false`) が含まれ、オリジンからのレスポンスにそのヘッダーが含まれているときの CloudFront の動作を決定します。

この設定が `true` に設定されていて、オリジンレスポンスにヘッダーが含まれている場合、CloudFront は、ビューワーに送信するレスポンスに対するポリシーのヘッダーを追加します。オリジンから受け取ったヘッダーは無視されます。

この設定が `false` に設定されていて、オリジンレスポンスでヘッダーが含まれている場合、CloudFront には、ビューワーに送信するレスポンスでオリジンから受信したヘッダーが含まれます。

オリジンレスポンスにヘッダーが含まれていない場合、CloudFront は、ビューワーに送信するレスポンスに対するポリシーのヘッダーを追加します。CloudFront は、この設定が `true` または `false` に設定されている場合に、この操作を実行します。

カスタムヘッダー

カスタムヘッダーの設定を使用して、レスポンスヘッダーポリシーでカスタム HTTP ヘッダーを追加および設定できます。CloudFront は、ビューワーに返すすべてのレスポンスに、これらのヘッダーを追加します。カスタムヘッダーごとにヘッダーの値も指定しますが、値の指定はオプションです。これは、CloudFront が値なしでレスポンスヘッダーを追加できるためです。

また各カスタムヘッダーには、独自のオリジンのオーバーライドの設定があります。

- この設定が `true` に設定されていて、オリジンレスポンスで、ポリシーにもカスタムヘッダーが含まれている場合、CloudFront は、ビューワーに送信するレスポンスに対するポリシーのカスタムヘッダーを追加します。オリジンから受け取ったヘッダーは無視されます。
- この設定が `false` であり、オリジンレスポンスで、ポリシーにカスタムヘッダーが含まれている場合、CloudFront には、ビューワーに送信するレスポンスでオリジンから受信したカスタムヘッダーが含まれます。
- オリジンレスポンスで、ポリシーにもカスタムヘッダーが含まれていない場合、CloudFront は、ビューワーに送信するレスポンスに対するポリシーのカスタムヘッダーを追加します。CloudFront は、この設定が `true` または `false` に設定されている場合に、この操作を実行します。

ヘッダーを削除

CloudFront がオリジンから受信したレスポンスから削除するヘッダーを指定し、これらのヘッダーを含まないレスポンスを CloudFront からビューワーに送信できます。CloudFront は、オブジェクトの提供元が CloudFront のキャッシュであるか、オリジンであるかにかかわらず、ビューワーに送信するすべてのレスポンスからヘッダーを削除します。例えば、ブラウザで不要になった `X-Powered-By` や `Vary` などのヘッダーを削除し、これらのヘッダーを削除したレスポンスを CloudFront からビューワーに送信できます。

レスポンスヘッダーポリシーを使用して削除するヘッダーを指定すると、CloudFront はまずヘッダーを削除し、次にレスポンスヘッダーポリシーの他のセクション (CORS ヘッダー、セキュリティヘッダー、カスタムヘッダーなど) で指定されているヘッダーを追加します。削除するヘッダーを指定しても、同じヘッダーをポリシーの別のセクションに追加している場合は、そのヘッダーが含まれたレスポンスが CloudFront からビューワーに送信されます。

> **Note**
>
> レスポンスヘッダーポリシーを使用して、CloudFront がオリジンから受信した `Server` ヘッダーと `Date` ヘッダーを削除し、これらの (オリジンから受信した) ヘッダーを含まないレスポンスを CloudFront からビューワーに送信できます。ただし、これを行うと、これらのヘッダーの CloudFront 独自のバージョンはビューワーに送信するレスポンスに追加されます。CloudFront が追加する `Server` ヘッダーの場合、ヘッダーの値は `CloudFront` です。

削除できないヘッダー

以下のヘッダーは、レスポンスヘッダーポリシーを使用して削除することはできません。これらのヘッダーをレスポンスヘッダーポリシーの [Remove headers] (ヘッダーを削除) セクション (API の `ResponseHeadersPolicyRemoveHeadersConfig`) で指定すると、エラーが表示されます。

- `Connection`
- `Content-Encoding`
- `Content-Length`
- `Expect`
- `Host`
- `Keep-Alive`
- `Proxy-Authenticate`

- Proxy-Authorization
- Proxy-Connection
- Trailer
- Transfer-Encoding
- Upgrade
- Via
- Warning
- X-Accel-Buffering
- X-Accel-Charset
- X-Accel-Limit-Rate
- X-Accel-Redirect
- X-Amz-Cf-.*
- X-Amzn-Auth
- X-Amzn-Cf-Billing
- X-Amzn-Cf-Id
- X-Amzn-Cf-Xff
- X-Amzn-ErrorType
- X-Amzn-Fle-Profile
- X-Amzn-Header-Count
- X-Amzn-Header-Order
- X-Amzn-Lambda-Integration-Tag
- X-Amzn-RequestId
- X-Cache
- X-Edge-.*
- X-Forwarded-Proto
- X-Real-Ip

Server-Timing ヘッダー

Server-Timing ヘッダー設定を使用して、CloudFront から送信された HTTP レスポンスの Server-Timing ヘッダーを有効にします。このヘッダーを使用して、CloudFront およびオリジン

の動作とパフォーマンスに関するインサイトを得るのに役立つメトリクスを表示することができます。例えば、キャッシュヒットを処理したキャッシュレイヤーを確認できます。または、キャッシュミスがある場合に、オリジンからの最初のバイトレイテンシーを確認できます。Server-Timing ヘッダーのメトリクスは、CloudFront またはオリジンの設定における問題のトラブルシューティングや効率のテストに役立ちます。

CloudFront で Server-Timing ヘッダーを使用する方法の詳細については、次のセクションを参照してください。

Server-Timing ヘッダーを有効にするには、レスポンスヘッダーポリシーを作成 (または編集) します。

トピック

- サンプリングレートと Pragma リクエストヘッダー
- オリジンからの Server-Timing ヘッダー
- Server-Timing ヘッダーのメトリクス
- Server-Timing ヘッダーの例

サンプリングレートと Pragma リクエストヘッダー

レスポンスヘッダーポリシーで Server-Timing ヘッダーを有効にするときは、サンプリングレートも指定します。サンプリングレートは、CloudFront が Server-Timing ヘッダーを追加するレスポンスの割合を指定する 0 ~ 100 の数値です。サンプリングレートを 100 に設定すると、CloudFront はレスポンスヘッダーポリシーがアタッチされているキャッシュ動作に一致するすべてのリクエストの HTTP レスポンスに、Server-Timing ヘッダーを追加します。50 に設定すると、CloudFront は、そのキャッシュ動作に一致するリクエストに対して、レスポンスの 50% にヘッダーを追加します。サンプリングレートは、0 ~ 100 の任意の数値 (小数点以下 4 桁まで) に設定できます。

サンプリングレートが 100 未満の数値に設定されている場合、CloudFront が Server-Timing ヘッダーを追加するレスポンスは制御できません。割合のみを制御できます。ただし、HTTP リクエストで値を server-timing に設定して Pragma ヘッダーを追加し、そのリクエストへのレスポンスで Server-Timing ヘッダーを受け取ることができます。これは、サンプリングレートの設定とは無関係に機能します。サンプリングレートがゼロ (0) に設定されている場合でも、リクエストに Pragma: server-timing ヘッダーが含まれていれば、CloudFront は Server-Timing ヘッダーをレスポンスに追加します。

オリジンからの Server-Timing ヘッダー

キャッシュミスがあり、CloudFront がリクエストをオリジンに転送すると、オリジンには、CloudFront へのレスポンスに `Server-Timing` ヘッダーが含まれている場合があります。この場合、CloudFront は、オリジンから受信した `Server-Timing` ヘッダーにメトリクスを追加します。CloudFront がビューワーに送信するレスポンスには、オリジンからの値と CloudFront が追加したメトリクスを含む単一の `Server-Timing` ヘッダーが含まれています。オリジンからのヘッダー値は、最後にある場合と、CloudFront がヘッダーに追加する 2 つのメトリクスセットの間にある場合があります。

キャッシュヒットが発生すると、CloudFront がビューワーに送信するレスポンスには、ヘッダー値に CloudFront メトリクスのみを含む単一の `Server-Timing` ヘッダーが含まれます (オリジンからの値は含まれません)。

Server-Timing ヘッダーのメトリクス

CloudFront が HTTP レスポンスに `Server-Timing` ヘッダーを追加する場合、ヘッダーの値には、CloudFront およびオリジンの動作とパフォーマンスに関するインサイトを得ることができる 1 つ以上のメトリクスが含まれます。次のリストには、すべてのメトリクスと想定される値が含まれています。`Server-Timing` ヘッダーには、CloudFront を介したリクエストとレスポンスの性質に応じて、これらのメトリクスの一部のみが含まれます。

これらのメトリクスのいくつかは、名前のみ (値なし) の `Server-Timing` ヘッダーに含まれます。その他は名前と価値です。メトリクスに値がある場合、名前と値はセミコロン (`;`) で区切られます。ヘッダーに複数のメトリクスが含まれている場合、メトリクスはカンマ (`,`) で区切られます。

cdn-cache-hit

CloudFront は、オリジンに対してリクエストを行わずに、キャッシュからレスポンスを提供しました。

cdn-cache-refresh

CloudFront は、リクエストをオリジンに送信してキャッシュされたオブジェクトがまだ有効であることを確認した後で、キャッシュからレスポンスを提供しました。この場合、CloudFront はオリジンから完全なオブジェクトを取得しませんでした。

cdn-cache-miss

CloudFront はキャッシュからレスポンスを提供しませんでした。この場合、CloudFront はレスポンスを返す前にオリジンから完全なオブジェクトをリクエストしました。

cdn-pop

CloudFront のどの Point of Presence (POP) がリクエストを処理したかを示す値が含まれます。

cdn-rid

CloudFront の一意のリクエスト識別子を示す値が含まれます。このリクエスト識別子 (RID) は、AWS Support に関する問題のトラブルシューティングに使用できます。

cdn-hit-layer

このメトリクスは、CloudFront がオリジンに対してリクエストを行わずにキャッシュからレスポンスを提供する場合に存在します。次のいずれかの値が含まれます。

- EDGE – CloudFront は POP ロケーションからキャッシュされたレスポンスを提供しました。
- REC – CloudFront は、リージョン別エッジキャッシュ (REC) ロケーションからキャッシュされたレスポンスを提供しました。
- オリジンシールド – CloudFront は、オリジンシールドとして機能している REC からキャッシュされたレスポンスを提供しました。

cdn-upstream-layer

このメトリクスは、CloudFront がオリジンに対して完全なオブジェクトをリクエストする場合に存在します。次のいずれかの値が含まれます。

- EDGE – POP ロケーションがリクエストをオリジンに直接送信しました。
- REC – REC ロケーションがリクエストをオリジンに直接送信しました。
- オリジンシールド – オリジンシールドとして機能している REC が、リクエストをオリジンに直接送信しました。

cdn-upstream-dns

オリジンの DNS レコードの取得にかかったミリ秒数を示す値が含まれます。値がゼロ (0) の場合、CloudFront がキャッシュされた DNS 結果を使用したか、既存の接続を再使用したことを示します。

cdn-upstream-connect

オリジン DNS リクエストが完了してから、オリジンへの TCP 接続 (および該当する場合は TLS 接続) が完了するまでのミリ秒数を示す値が含まれます。値がゼロ (0) の場合、CloudFront が既存の接続を再使用したことを示します。

cdn-upstream-fbl

オリジン HTTP リクエストが完了してから、オリジンからのレスポンスで最初のバイトを受信するまでのミリ秒数 (最初のバイトレイテンシー) を示す値が含まれます。

cdn-downstream-fbl

エッジロケーションがリクエストの受信を終了してから、レスポンスの最初のバイトをビューワーに送信するまでのミリ秒を示す値が含まれます。

Server-Timing ヘッダーの例

`Server-Timing` ヘッダー設定が有効になっているときに、ビューワーが CloudFront から受信する可能性がある `Server-Timing` の例を以下に示します。

Example – キャッシュミス

リクエストされたオブジェクトが CloudFront キャッシュ内にないときにビューワーが受け取る可能性のある `Server-Timing` ヘッダーの例を以下に示します。

```
Server-Timing: cdn-upstream-layer;desc="EDGE",cdn-upstream-dns;dur=0,cdn-upstream-
connect;dur=114,cdn-upstream-fbl;dur=177,cdn-cache-miss,cdn-pop;desc="PHX50-C2",cdn-
rid;desc="yNPsyYn7skvTzwWkq3Wcc8Nj_foxUjQUe9H1ifslzWhb0w7aLbFvGg==",cdn-downstream-
fbl;dur=436
```

この `Server-Timing` ヘッダーは以下を示します。

- オリジンリクエストが CloudFront Point of Presence (POP) ロケーション (`cdn-upstream-layer;desc="EDGE"`) から送信されたこと。
- CloudFront はキャッシュされた DNS 結果をオリジン (`cdn-upstream-dns;dur=0`) に使用しました。
- CloudFront がオリジン(`cdn-upstream-connect;dur=114`) への TCP (および該当する場合は TLS) 接続を完了するまでに 114 ミリ秒かかったこと。
- リクエスト (`cdn-upstream-fbl;dur=177`) の完了後、CloudFront がオリジンからレスポンスの最初のバイトを受信するまでに 177 ミリ秒かかったこと。
- リクエストされたオブジェクトは CloudFront のキャッシュ (`cdn-cache-miss`) にありませんでした。
- リクエストは、コード `PHX50-C2` (`cdn-pop;desc="PHX50-C2"`) によって識別されるエッジロケーションで受信されました。

- このリクエストの CloudFront の一意の ID は yNPsyYn7skvTzwWkq3Wcc8Nj_foxUjQUe9H1ifslzWhb0w7aLbFvGg== (cdn-rid;desc="yNPsyYn7skvTzwWkq3Wcc8Nj_foxUjQUe9H1ifslzWhb0w7aLbFvGg==") でした。
- ビューワーリクエストを受け取った後、CloudFront がレスポンスの最初のバイトをビューワーに送信するのに 436 ミリ秒かかったこと (cdn-downstream-fbl;dur=436)。

Example – キャッシュヒット

リクエストされたオブジェクトが CloudFront キャッシュ内にあるときにビューワーが受け取る可能性のある Server-Timing ヘッダーの例を以下に示します。

```
Server-Timing: cdn-cache-hit,cdn-pop;desc="SEA19-C1",cdn-
rid;desc="nQBz4aJU2kP9iC3KHEq7vFxfMozu-VYBwGzkW9diOpeVc7xsrLKj-g==",cdn-hit-
layer;desc="REC",cdn-downstream-fbl;dur=137
```

この Server-Timing ヘッダーは以下を示します。

- リクエストされたオブジェクトはキャッシュ (cdn-cache-hit) 内にありました。
- リクエストは、コード SEA19-C1 (cdn-pop;desc="SEA19-C1") によって識別されるエッジロケーションで受信されました。
- このリクエストの CloudFront の一意の ID は nQBz4aJU2kP9iC3KHEq7vFxfMozu-VYBwGzkW9diOpeVc7xsrLKj-g== (cdn-rid;desc="nQBz4aJU2kP9iC3KHEq7vFxfMozu-VYBwGzkW9diOpeVc7xsrLKj-g==") でした。
- リクエストされたオブジェクトは、リージョン別エッジキャッシュ (REC) ロケーション (cdn-hit-layer;desc="REC") にキャッシュされました。
- ビューワーリクエストを受け取った後、CloudFront がレスポンスの最初のバイトをビューワーに送信するのに 137 ミリ秒かかったこと (cdn-downstream-fbl;dur=137)。

レスポンスヘッダーポリシーの作成

レスポンスヘッダーポリシーを使用して、Amazon CloudFront が HTTP レスポンスで追加または削除する HTTP ヘッダーを指定できます。レスポンスヘッダーポリシーおよびそれらを使用する理由の詳細については、「ポリシーを使用してレスポンスヘッダーを追加または削除する」を参照してください。

レスポンスヘッダーポリシーは、CloudFront コンソールで作成できます。または、AWS CloudFormation、AWS Command Line Interface (AWS CLI)、CloudFront API のいずれかを使用して作成することもできます。レスポンスヘッダーポリシーを作成したら、CloudFront ディストリビューションの 1 つ以上のキャッシュ動作にアタッチします。

カスタムレスポンスヘッダーポリシーを作成する前に、マネージドレスポンスヘッダーポリシーの 1 つがユースケースに適合するかどうか確認します。適合する場合は、そのポリシーをキャッシュ動作にアタッチできます。これにより、独自のレスポンスヘッダーポリシーを作成したり管理したりする必要はありません。

Console

レスポンスヘッダーポリシーを作成するには (コンソール)

1. AWS Management Console にサインインしてから、https://console.aws.amazon.com/cloudfront/v4/home#/policies/responseHeaders の CloudFront コンソールの [Policies] (ポリシー) ページで [Response headers] (レスポンスヘッダー) タブに行きます。
2. [Create response headers policy] (レスポンスヘッダーポリシーの作成) を選択します。
3. [Create response headers policy] (レスポンスヘッダーポリシーの作成) フォームで、次の操作を行います。

   a. [Details] (詳細) パネルで、レスポンスヘッダーポリシーおよび (任意で) ポリシーの目的を説明する [Description] (説明) に「名前」を入力します。

   b. [Cross-Origin Resource Sharing (CORS)] パネルで、[Configure CORS] (CORS を設定する) トグルを選択し、ポリシーに追加する CORS ヘッダーを設定します。CloudFront がオリジンから受信するヘッダーを上書きするためにヘッダーを設定する場合は、[Origin override] (オリジンの上書き) チェックボックスをオンにします。

      CORS ヘッダー設定の詳細については、「the section called “CORS ヘッダー”」を参照してください。

   c. [Security headers] (セキュリティヘッダー) パネルで、トグルを選択し、ポリシーに追加する各セキュリティヘッダーを設定します。

      セキュリティヘッダー設定の詳細については、「the section called “セキュリティヘッダー”」を参照してください。

   d. [Custom headers] (カスタムヘッダー) パネルで、ポリシーに含めるカスタムヘッダーを追加します。

カスタムヘッダー設定の詳細については、「the section called “カスタムヘッダー”」を参照してください。

e. [Remove headers] (ヘッダーの削除) パネルで、CloudFront でオリジンのレスポンスから削除し、CloudFront からビューワーに送信するレスポンスには含めないヘッダーの名前を追加します。

ヘッダーの削除設定の詳細については、「the section called “ヘッダーを削除”」を参照してください。

f. [Server-Timing header] (Server-Timing ヘッダー) パネルで、[Enable] (有効化) トグルを選択して、サンプリングレート (0 ~ 100 の数値) を入力します。

`Server-Timing` ヘッダーの詳細については、「the section called “Server-Timing ヘッダー”」を参照してください。

4. [Create] (作成) を選択して、ポリシーを作成します。

レスポンスヘッダーポリシーを作成したら、CloudFront ディストリビューションのキャッシュ動作にアタッチできます。

既存のディストリビューションにレスポンスヘッダーポリシーをアタッチするには (コンソール)

1. https://console.aws.amazon.com/cloudfront/v4/home#/distributions で CloudFront コンソールの [Distributions] (ディストリビューション) ページを開きます。
2. 更新するディストリビューションを選択し、[動作] タブを選択します。
3. 更新するキャッシュ動作を選び、[Edit] (編集) を選択します。

   または、新しいキャッシュ動作を作成するには、[動作を作成] を選択します。
4. [Response headers policy] (レスポンスヘッダーポリシー) の場合は、キャッシュ動作に追加するポリシーを選択します。
5. [Save changes] (変更を保存) を選択して、キャッシュ動作を更新します。新しいキャッシュ動作を作成する場合は、[Create behavior] (動作を作成) を選択します。

新しいディストリビューションにレスポンスヘッダーポリシーをアタッチするには (コンソール)

1. https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます
2. [ディストリビューションの作成] を選択します。

3. [Response headers policy] (レスポンスヘッダーポリシー) の場合は、キャッシュ動作に追加するポリシーを選択します。
4. その他のディストリビューションの設定を選択します。詳細については、「the section called “ディストリビューションの設定”」を参照してください。
5. [Create distribution] (ディストリビューションを作成) を選択して、ディストリビューションを作成します。

AWS CloudFormation

AWS CloudFormation を使用してレスポンスヘッダーポリシーを作成するには、`AWS::CloudFront::ResponseHeadersPolicy` リソースタイプを使用します。以下に例を示します。AWS CloudFormationレスポンスヘッダーポリシーを作成するための YAML 形式のテンプレート構文。

```
Type: AWS::CloudFront::ResponseHeadersPolicy
Properties:
  ResponseHeadersPolicyConfig:
    Name: EXAMPLE-Response-Headers-Policy
    Comment: Example response headers policy for the documentation
    CorsConfig:
      AccessControlAllowCredentials: false
      AccessControlAllowHeaders:
        Items:
          - '*'
      AccessControlAllowMethods:
        Items:
          - GET
          - OPTIONS
      AccessControlAllowOrigins:
        Items:
          - https://example.com
          - https://docs.example.com
      AccessControlExposeHeaders:
        Items:
          - '*'
      AccessControlMaxAgeSec: 600
      OriginOverride: false
    CustomHeadersConfig:
      Items:
        - Header: Example-Custom-Header-1
          Value: value-1
```

```
          Override: true
        - Header: Example-Custom-Header-2
          Value: value-2
          Override: true
    SecurityHeadersConfig:
      ContentSecurityPolicy:
        ContentSecurityPolicy: default-src 'none'; img-src 'self'; script-src
 'self'; style-src 'self'; object-src 'none'; frame-ancestors 'none'
        Override: false
      ContentTypeOptions: # You don't need to specify a value for 'X-Content-Type-
Options'.
                          # Simply including it in the template sets its value to
 'nosniff'.
        Override: false
      FrameOptions:
        FrameOption: DENY
        Override: false
      ReferrerPolicy:
        ReferrerPolicy: same-origin
        Override: false
      StrictTransportSecurity:
        AccessControlMaxAgeSec: 63072000
        IncludeSubdomains: true
        Preload: true
        Override: false
      XSSProtection:
        ModeBlock: true # You can set ModeBlock to 'true' OR set a value for
 ReportUri, but not both
        Protection: true
        Override: false
    ServerTimingHeadersConfig:
      Enabled: true
      SamplingRate: 50
    RemoveHeadersConfig:
      Items:
        - Header: Vary
        - Header: X-Powered-By
```

詳細については、AWS CloudFormationユーザーガイド内の「AWS:: CloudFront:: ResponseHeadersPolicy」を参照してください。

CLI

AWS Command Line Interface (AWS CLI) でレスポンスヘッダーポリシーを作成するには、aws cloudfront create-response-headers-policy コマンドを使用します。コマンドの入力パラメータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

レスポンスヘッダーポリシーを作成するには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、response-headers-policy.yaml という名前のファイルを作成します。このファイルには、create-response-headers-policy コマンドのすべての入力パラメータが含まれます。

```
aws cloudfront create-response-headers-policy --generate-cli-skeleton yaml-input
 > response-headers-policy.yaml
```

2. 先ほど作成した response-headers-policy.yaml ファイルを開きます。ファイルを編集してポリシー名と必要なレスポンスヘッダーポリシー設定を指定し、ファイルを保存します。

   レスポンスヘッダーポリシー設定の詳細については、「the section called “レスポンスヘッダーポリシーを理解する”」を参照してください。

3. 次のコマンドを使用してレスポンスヘッダーポリシーを作成します。作成するポリシーでは、response-headers-policy.yaml ファイルからの入力パラメータを使用します。

```
aws cloudfront create-response-headers-policy --cli-input-yaml file://response-
headers-policy.yaml
```

   コマンド出力の Id 値を書き留めます。これはレスポンスヘッダーポリシー ID です。ポリシーを CloudFront ディストリビューションのキャッシュ動作にアタッチするために必要です。

既存のディストリビューションにレスポンスヘッダーポリシーをアタッチするには (入力ファイルを含む CLI)

1. 以下のコマンドを使用して、更新する CloudFront ディストリビューションのディストリビューション設定を保存します。*distribution_ID* をディストリビューション ID に置き換えます。

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml >
 dist-config.yaml
```

2. 先ほど作成した dist-config.yaml という名前のファイルを開きます。ファイルを編集し、レスポンスヘッダーポリシーを使用するように、次の変更をキャッシュ動作に加えます。

   - キャッシュ動作で、ResponseHeadersPolicyId という名前のフィールドを追加します。フィールドの値には、ポリシーの作成後に書き留めたレスポンスヘッダーポリシー ID を使用します。
   - ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。

   完了したら、ファイルを保存します。

3. レスポンスヘッダーポリシーを使用するようにディストリビューションを更新するには、次のコマンドを使用します。*distribution_ID* をディストリビューション ID に置き換えます。

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

新しいディストリビューションにレスポンスヘッダーポリシーをアタッチするには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、distribution.yaml という名前のファイルを作成します。このファイルには、create-distribution コマンドのすべての入力パラメータが含まれます。

```
aws cloudfront create-distribution --generate-cli-skeleton yaml-input >
 distribution.yaml
```

2. 先ほど作成した distribution.yaml ファイルを開きます。デフォルトのキャッシュ動作の [ResponseHeadersPolicyId] フィールドに、ポリシーの作成後に書き留めたレスポンスヘッダーポリシー ID を入力します。ファイルの編集を続行して必要なディストリビューション設定を指定し、完了したらファイルを保存します。

ディストリビューション設定の詳細については、「ディストリビューション設定リファレンス」を参照してください。

3. 次のコマンドを使用して、distribution.yaml ファイルの入力パラメータを使用し、ディストリビューションを作成します。

```
aws cloudfront create-distribution --cli-input-yaml file://distribution.yaml
```

API

CloudFront API を使用してレスポンスヘッダーポリシーを作成するには、「CreateResponseHeadersPolicy」を使用します。この API コールで指定するフィールドの詳細については、「the section called “レスポンスヘッダーポリシーを理解する”」、および AWS SDK やその他の API クライアントの API リファレンスドキュメントを参照してください。

レスポンスヘッダーポリシーを作成したら、次の API コールのいずれかを使用して、それをキャッシュ動作にアタッチできます。

- 既存のディストリビューションのキャッシュ動作にアタッチするには、UpdateDistribution を使用します。
- 新しいディストリビューションのキャッシュ動作にアタッチするには、CreateDistribution を使用します。

これらの API コールの両方について、キャッシュ動作内で、ResponseHeadersPolicyId フィールドにレスポンスヘッダーポリシー ID を指定します。これらの API コールで指定するその他フィールドの詳細については、「ディストリビューション設定リファレンス」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

マネージドレスポンスヘッダーポリシーを使用する

レスポンスヘッダーポリシーを使用して、Amazon CloudFront からビューワーに送信するレスポンスで削除または追加する HTTP ヘッダーを指定できます。レスポンスヘッダーポリシーおよびそれらを使用する理由の詳細については、「ポリシーを使用してレスポンスヘッダーを追加または削除する」を参照してください。

CloudFront で、CloudFront ディストリビューションのキャッシュ動作にアタッチできるマネージドレスポンスヘッダーポリシーを提供します。マネージドレスポンスヘッダーポリシーを使用すると、独自のポリシーを記述したり、維持したりする必要がありません。このマネージドポリシーには、一般的ユースケース用の HTTP レスポンスヘッダーのセットが含まれています。

マネージドレスポンスヘッダーポリシーを使用するには、ディストリビューションのキャッシュ動作にそのポリシーをアタッチします。このプロセスは、カスタムレスポンスヘッダーポリシーを作成するときと同じです。ただし、新しいポリシーを作成する代わりに、マネージドポリシーの 1 つをアタッチします。ポリシーは、名前 (コンソールを使用) または ID (AWS CloudFormation、AWS CLI、または AWS SDK を使用) を使用してアタッチします。名前と ID は、次のセクションに記載されています。

詳細については、「the section called “レスポンスヘッダーポリシーの作成”」を参照してください。

次のトピックでは、使用可能なマネージドレスポンスヘッダーポリシーについて説明します。

トピック

CORS-and-SecurityHeadersPolicy

このポリシーを CloudFront コンソールで見る

このマネージドポリシーを使用して、オリジンからの単一の CORS リクエストが許可されます。このポリシーはまた、CloudFront がビューワーに送信するすべてのレスポンスに、セキュリティヘッダーのセットを追加します。このポリシーによって、the section called “SimpleCORS” ポリシーおよび the section called “SecurityHeadersPolicy” ポリシーを 1 つにまとめます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
e61eb60c-9c35-4d20-a928-2b84e02af89c
```

ポリシー設定

	ヘッダー名	ヘッダー値	オーバーライドオリジンですか。
CORS ヘッダー:	`Access-Control-Allow-Origin`	`*`	いいえ
セキュリティヘッダー:	`Referrer-Policy`	`strict-origin-when-cross-origin`	いいえ
	`Strict-Transport-Security`	`max-age=31536000`	いいえ
	`X-Content-Type-Options`	`nosniff`	はい
	`X-Frame-Options`	`SAMEORIGIN`	いいえ
	`X-XSS-Protection`	`1; mode=block`	いいえ

CORS-With-Preflight

このポリシーを CloudFront コンソールで見る

このマネージドポリシーを使用して、プリフライトリクエストを含むオリジンからの CORS リクエストが許可されます。プリフライトリクエスト (HTTP の `OPTIONS` メソッドを使用) の場合、CloudFront は次の 3 つのヘッダーすべてをレスポンスに追加します。単一の CORS リクエストの場合、CloudFront は `Access-Control-Allow-Origin` ヘッダーのみ追加します。

CloudFront がオリジンから受信するレスポンスにこれらのヘッダーが含まれる場合、CloudFront はビューワーへのレスポンスに、受信したヘッダー (およびその値) を使用します。CloudFront はこのポリシーのヘッダーを使用しません。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

`5cc3b908-e619-4b99-88e5-2cf7f45965bd`

ポリシー設定

	ヘッダー名	ヘッダー値	オーバーライドオリジンですか。
CORS ヘッダー:	`Access-Control-Allow-Methods`	`DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT`	いいえ
	`Access-Control-Allow-Origin`	`*`	
	`Access-Control-Expose-Headers`	`*`	

CORS-with-preflight-and-SecurityHeadersPolicy

このポリシーを CloudFront コンソールで見る

このマネージドポリシーを使用して、オリジンからの CORS リクエストが許可されます。これには、プリフライトリクエストが含まれます。このポリシーはまた、CloudFront がビューワーに送信するすべてのレスポンスに、セキュリティヘッダーのセットを追加します。このポリシーによって、the section called “CORS-With-Preflight” ポリシーおよび the section called “SecurityHeadersPolicy” ポリシーを 1 つにまとめます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
eaab4381-ed33-4a86-88ca-d9558dc6cd63
```

ポリシー設定

	ヘッダー名	ヘッダー値	オーバーライドオリジンですか。
CORS ヘッダー:	Access-Control-Allow-Methods	DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT	いいえ
	Access-Control-Allow-Origin	*	
	Access-Control-Expose-Headers	*	
セキュリティヘッダー:	Referrer-Policy	strict-origin-when-cross-origin	いいえ
	Strict-Transport-Security	max-age=31536000	いいえ
	X-Content-Type-Options	nosniff	はい
	X-Frame-Options	SAMEORIGIN	いいえ
	X-XSS-Protection	1; mode=block	いいえ

SecurityHeadersPolicy

このポリシーを CloudFront コンソールで見る

このマネージドポリシーを使用して、CloudFront がビューワーに送信するすべてのレスポンスに、セキュリティヘッダーのセットを追加します。これらのセキュリティヘッダーの詳細については、Mozilla のウェブセキュリティに関するガイドラインを参照してください。

このレスポンスヘッダーポリシーでは、CloudFront はすべてのレスポンスに X-Content-Type-Options: nosniff を追加します。これは、CloudFront がオリジンから受け取ったレスポンス

にこのヘッダーが含まれていなかった場合です。このポリシーのその他すべてのヘッダーについては、CloudFront がオリジンから受信するレスポンスにこのヘッダーが含まれる場合、CloudFront はビューワーへのレスポンスに、受信したヘッダー (およびその値) を使用します。CloudFront はこのポリシーのヘッダーを使用しません。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

`67f7725c-6f97-4210-82d7-5512b31e9d03`

ポリシー設定

	ヘッダー名	ヘッダー値	オーバーライドオリジンですか。
セキュリティヘッダー:	`Referrer-Policy`	`strict-origin-when-cross-origin`	いいえ
	`Strict-Transport-Security`	`max-age=31536000`	いいえ
	`X-Content-Type-Options`	`nosniff`	はい
	`X-Frame-Options`	`SAMEORIGIN`	いいえ
	`X-XSS-Protection`	`1; mode=block`	いいえ

SimpleCORS

このポリシーを CloudFront コンソールで見る

このマネージドポリシーを使用して、オリジンからの単一の CORS リクエストが許可されます。このポリシーを使用して、CloudFront は、`Access-Control-Allow-Origin: *`単一の CORS リクエストのすべてのレスポンスにヘッダーを追加します。

CloudFront がオリジンから受信するレスポンスに `Access-Control-Allow-Origin` ヘッダーが含まれる場合、CloudFront は、ビューワーへのレスポンスにそのヘッダー (およびその値) を使用します。CloudFront はこのポリシーのヘッダーを使用しません。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
60669652-455b-4ae9-85a4-c4c02393f86c
```

ポリシー設定

	ヘッダー名	ヘッダー値	オーバーライドオリジンですか。
CORS ヘッダー:	`Access-Control-Allow-Origin`	*	いいえ

リクエストとレスポンスの動作

以下のセクションでは、CloudFront がビューワーリクエストを処理してそのリクエストを Amazon S3 またはカスタムオリジンに転送する方法、および CloudFront がオリジンからのレスポンスを処理する方法 (CloudFront が 4xx および 5xx HTTP ステータスコードを処理およびキャッシュする方法を含む) について説明します。

トピック

CloudFront が HTTP および HTTPS リクエストを処理する方法

Amazon S3 オリジンの場合、デフォルトでは、CloudFront は、HTTP および HTTPS プロトコルで、CloudFront ディストリビューション内のオブジェクトのリクエストを受け取ります。次に、CloudFront は、リクエストと同じプロトコルを使用して、リクエストを Amazon S3 バケットに転送します。

カスタムオリジンでは、ディストリビューションを作成する場合、CloudFront がオリジンにアクセスする方法を指定できます (HTTP のみか、ビューワーが使用しているプロトコルと一致させます)。カスタムオリジンにおいて CloudFront が HTTP および HTTPS リクエストを処理する方法については、「プロトコル」を参照してください。

エンドユーザーが HTTPS を使用してのみオブジェクトにアクセスできるようにディストリビューションを制限する方法については、「CloudFront で HTTPS を使用する」を参照してください。

> **Note**
>
> HTTPS リクエストの料金は HTTP リクエストの料金よりも高くなります。請求料率の詳細については、「CloudFront の料金」を参照してください。

Amazon S3 オリジンに対するリクエストとレスポンスの動作

オリジンとして Amazon S3 を使用している場合に、CloudFront がリクエストとレスポンスを処理する方法については、以下のセクションを参照してください。

トピック

- CloudFront がリクエストを処理して Amazon S3 オリジンに転送する方法
- CloudFront が Amazon S3 オリジンからのレスポンスを処理する方法

CloudFront がリクエストを処理して Amazon S3 オリジンに転送する方法

CloudFront がビューワーのリクエストを処理して Amazon S3 オリジンに転送する方法について説明します。

目次

- キャッシュ期間および最小 TTL
- クライアント IP アドレス
- 条件付き GET リクエスト
- cookie
- クロスオリジンリソース共有 (CORS)
- 本文が含まれている GET リクエスト
- HTTP メソッド
- CloudFront が削除または更新する HTTP リクエストヘッダー
- リクエストの最大長と URL の最大長
- OCSP Stapling
- プロトコル
- クエリ文字列
- オリジン接続のタイムアウトと試行

- オリジン応答タイムアウト
- 同一オブジェクトへの同時リクエスト (リクエストを折りたたむ)

キャッシュ期間および最小 TTL

CloudFront が別のリクエストをオリジンに転送するまでにオブジェクトを CloudFront キャッシュに保持する時間を制御するには、次の手順を実行します。

- `Cache-Control` または `Expires` ヘッダーフィールドを各オブジェクトに追加するようにオリジンを構成します。
- CloudFront キャッシュ動作で、最小 TTL の値を指定します。
- デフォルト値の 24 時間を使用します。

詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

クライアント IP アドレス

ビューワーが CloudFront に送信したリクエストに、`X-Forwarded-For` リクエストヘッダーを含めなかった場合、CloudFront は TCP 接続からビューワーの IP アドレスを取得して、IP アドレスを含めた `X-Forwarded-For` ヘッダーを追加し、リクエストをオリジンに転送します。たとえば、CloudFront が TCP 接続から IP アドレス `192.0.2.2` を取得する場合、以下のヘッダーをオリジンに転送します。

```
X-Forwarded-For: 192.0.2.2
```

ビューワーがリクエストを CloudFront に転送して `X-Forwarded-For` リクエストヘッダーを含める場合、CloudFront はビューワーの IP アドレスを TCP 接続から取得してそれを `X-Forwarded-For` ヘッダーの末尾に追加し、リクエストをオリジンに転送します。たとえば、ビューワーのリクエストに `X-Forwarded-For: 192.0.2.4,192.0.2.3` が含まれ、CloudFront が TCP 接続から IP アドレス `192.0.2.2` を取得する場合、以下のヘッダーをオリジンに転送します。

```
X-Forwarded-For: 192.0.2.4,192.0.2.3,192.0.2.2
```

> **Note**
>
> `X-Forwarded-For` ヘッダーには、IPv4 アドレス (192.0.2.44 など) および IPv6 アドレス (2001:0db8:85a3::8a2e:0370:7334 など) が含まれます。

条件付き GET リクエスト

CloudFront は、エッジキャッシュで有効期限切れになっているオブジェクトに対するリクエストを受け取ると、リクエストを Amazon S3 オリジンに転送し、オブジェクトの最新バージョンを取得するか、CloudFront エッジキャッシュに最新バージョンが既に存在することを Amazon S3 に確認します。Amazon S3 はオブジェクトを CloudFront に最初に送信するときに、`ETag` 値と `LastModified` 値をレスポンスに含めます。CloudFront は、Amazon S3 に転送する新しいリクエストに、以下のヘッダーのどちらかまたは両方を追加します。

- オブジェクトの有効期限切れバージョンの `If-Match` 値が含まれる `If-None-Match` または `ETag` ヘッダー。
- オブジェクトの有効期限切れバージョンの `If-Modified-Since` 値が含まれる `LastModified` ヘッダー。

Amazon S3 は、この情報を使用して、オブジェクトが更新されているかどうかを判別します。つまり、オブジェクト全体を CloudFront に返すか、または HTTP 304 ステータスコード (変更なし) のみを返すかを判別します。

cookie

Amazon S3 は Cookie を処理しません。Cookie を Amazon S3 オリジンに転送するようにキャッシュ動作を構成した場合、CloudFront は Cookie を転送しますが、Amazon S3 は Cookie を無視します。同じオブジェクトに対する今後すべてのリクエストは、Cookie を変更するかどうかに関わらず、キャッシュ内の既存のオブジェクトから処理されます。

クロスオリジンリソース共有 (CORS)

CloudFront で Amazon S3 の Cross-Origin Resource Sharing 設定を尊重する場合は、選択したヘッダーを Amazon S3 に転送するように CloudFront を設定します。詳細については、「リクエストヘッダーに基づいてコンテンツをキャッシュする」を参照してください。

本文が含まれている GET リクエスト

ビューワー `GET` のリクエストの本文が含まれている場合、CloudFront はビューワーに HTTP ステータスコード 403 (禁止) を返します。

HTTP メソッド

サポートするすべての HTTP メソッドを処理するよう CloudFront を構成すると、CloudFront はビューワーからの以下のリクエストを受け入れて Amazon S3 オリジンに転送します。

- DELETE
- GET
- HEAD
- OPTIONS
- PATCH
- POST
- PUT

CloudFront は、GET リクエストと HEAD リクエストへの応答を常にキャッシュします。OPTIONS リクエストへの応答をキャッシュするように CloudFront を設定することもできます。CloudFront は他のメソッドを使用するリクエストへのレスポンスをキャッシュしません。

マルチパートアップロードを使用してオブジェクトを Amazon S3 バケットに追加する場合は、CloudFront オリジンアクセスコントロール (OAC) をディストリビューションに追加して、その OAC に必要な許可を付与する必要があります。詳細については、「the section called “Amazon Simple Storage Service オリジンへのアクセスを制限する”」を参照してください。

> **Important**
>
> CloudFront がサポートするすべての HTTP メソッドを受け入れて Amazon S3 に転送するように CloudFront を設定する場合、Amazon S3 コンテンツへのアクセスを制限する CloudFront OAC を作成して、OAC に必要なアクセス許可を付与する必要があります。例えば、PUT を使用するために、上記のメソッドを受け入れて転送するように CloudFront を設定する場合は、削除すべきでないリソースをビューワーが削除できないようにするために、DELETE リクエストを適切に処理するように Amazon S3 バケットポリシーを設定する必要があります。詳細については、「the section called “Amazon Simple Storage Service オリジンへのアクセスを制限する”」を参照してください。

Amazon S3 がサポートする操作の詳細については、「Amazon S3 ドキュメント」を参照してください。

CloudFront が削除または更新する HTTP リクエストヘッダー

CloudFront は、リクエストを Amazon S3 オリジンに転送する前に、一部のヘッダーを削除または更新します。ほとんどのヘッダーで、この動作はカスタムオリジンの場合と同じです。すべての HTTP

リクエストヘッダーの一覧と、CloudFront がそれを処理する方法については、「HTTP リクエストヘッダーと CloudFront の動作 (カスタムオリジンおよび Amazon S3 オリジン)」を参照してください。

リクエストの最大長と URL の最大長

パス、クエリ文字列 (ある場合)、ヘッダーを含め、リクエストの最大長は 20480 バイトです。

CloudFront はリクエストから URL を構築します。この URL の最大長は 8192 文字です。

リクエストまたは URL が最大長を超えると、CloudFront は、HTTP ステータスコード 413 (Request Entity Too Large) をビューワーに返し、ビューワーへの TCP 接続を終了します。

OCSP Stapling

ビューワーがオブジェクトへの HTTPS リクエストを送信する場合、CloudFront またはビューワーは、ドメインの SSL 証明書が無効になっていないことを認証機関 (CA) に確認する必要があります。OCSP Stapling を使用すると、CloudFront で証明書を検証して CA からの応答をキャッシュできるため、クライアントが直接 CA に対して証明書を検証する必要がなくなり、証明書の検証速度が向上します。

同一ドメイン内のオブジェクトに対する多数の HTTPS リクエストを CloudFront が受信した場合は、OCSP Stapling によるパフォーマンス向上がさらに顕著になります。CloudFront エッジロケーション内の各サーバーは、別々の検証リクエストを送信する必要があります。CloudFront が同一ドメインで多数の HTTPS リクエストを受信すると、エッジロケーション内のすべてのサーバーは、CA からの即座のレスポンスに応じて、SSL ハンドシェイクのパケットにステープルできます。証明書が有効であることをビューワーが確認すると、CloudFront はリクエストされたオブジェクトを提供できます。CloudFront エッジロケーション内でディストリビューションが十分なトラフィックを確保できない場合、新しいリクエストは、CA に対して証明書がまだ検証されていないサーバーに誘導される可能性が高くなります。この場合は、ビューワーが検証ステップを別途実行し、CloudFront サーバーがオブジェクトを提供します。この CloudFront サーバーも CA に検証リクエストを送信するため、同じドメイン名が含まれるリクエストを次に受信したときには、CA からの検証応答が既に存在しているということになります。

プロトコル

CloudFront は、ビューワーリクエストのプロトコル (HTTP または HTTPS) に基づいて、HTTP または HTTPS リクエストをオリジンサーバーに転送します。

> **Important**
>
> Amazon S3 バケットがウェブサイトエンドポイントとして設定されている場合、オリジンとの通信に HTTPS を使用するように CloudFront を設定することはできません。Amazon S3 はその設定で HTTPS 接続をサポートしていないためです。

クエリ文字列

CloudFront でクエリ文字列パラメータを Amazon S3 オリジンに転送するかどうかを設定できます。詳細については、「クエリ文字列パラメータに基づいてコンテンツをキャッシュする」を参照してください。

オリジン接続のタイムアウトと試行

オリジン接続タイムアウトは、オリジンへの接続を確立しようとしたときに CloudFront が待機する秒数です。

オリジン接続試行は、CloudFront がオリジンへの接続を試行する回数です。

これらの設定により、セカンダリオリジンにフェイルオーバーするか (オリジングループの場合)、ビューワーにエラーレスポンスを返す前に、CloudFront がオリジンへの接続を試行する時間が決まります。デフォルトでは、CloudFront はセカンダリオリジンへの接続を試行したり、エラーレスポンスを返したりする前に 30 秒 (それぞれ 10 秒間の試行が 3 回) 待機します。接続タイムアウトを短くするか、試行回数を減らすか、その両方を行うことで、この時間を短縮できます。

詳細については、「オリジンのタイムアウトと試行を制御する」を参照してください。

オリジン応答タイムアウト

オリジン応答タイムアウト (オリジンの読み取りタイムアウトまたはオリジンリクエストタイムアウトとも呼ばれます) は、次の両方に適用されます。

- CloudFront がリクエストをオリジンに転送してからレスポンスを受け取るまでの待機時間 (秒)。
- CloudFront がオリジンからレスポンスのパケットを受け取ってから次のパケットを受け取るまでの待機時間 (秒)。

CloudFront の動作は、ビューワーリクエストの HTTP メソッドによって決まります。

- GET および HEAD リクエスト – オリジンが 30 秒以内に応答しない場合、または 30 秒間応答を停止する場合は、CloudFront は接続を中断します。指定されたオリジン接続の試行回数が 1 回を超える場合、CloudFront は完全な応答の取得を再試行します。オリジン接続の試行回数設定の値で決められているように、CloudFront は最大 3 回試行します。最後の試行でもオリジンが応答しない場合、CloudFront は同じオリジンのコンテンツに対する別のリクエストを受け取るまで接続を試みません。
- DELETE、OPTIONS、PATCH、PUT、POST リクエスト – オリジンが 30 秒以内に応答しない場合、CloudFront は接続を中断し、オリジンへの接続を再試行しません。クライアントは、必要に応じてリクエストを再送信できます。

Amazon S3 オリジン (静的ウェブサイトホスティングで設定されていない S3 バケット) の応答タイムアウトを変更することはできません。

同一オブジェクトへの同時リクエスト (リクエストを折りたたむ)

CloudFront エッジロケーションがオブジェクトのリクエストを受け取り、オブジェクトが現在キャッシュにないか、有効期限が切れている場合、CloudFront はすぐにオリジンにリクエストを送信します。ただし、同一オブジェクトへの同時リクエストがある場合 (同じキャッシュキーを使用)、CloudFront が最初のリクエストへのレスポンスを受信する前に、同一オブジェクト (同じキャッシュキー) への追加のリクエストがエッジロケーションに届く場合、CloudFront は一時停止してから、追加のリクエストをオリジンに転送します。この短い一時停止により、オリジンでの負荷を減らすことができます。CloudFront は、一時停止中に受け取ったすべてのリクエストに、元のリクエストからのレスポンスを送信します。これはリクエストの折りたたみと呼ばれます。CloudFront ログでは、最初のリクエストは `x-edge-result-type` フィールドで `Miss` と識別され、折りたたまれたリクエストは `Hit` とし識別されます。CloudFront ログの詳細については、「the section called “CloudFront とエッジ関数の口グ記録”」を参照してください。

CloudFront は、キャッシュキーを共有するリクエストのみを折りたたみます。リクエストヘッダーまたはクッキーまたはクエリ文字列に基づいてキャッシュするように CloudFront を設定した場合など、追加のリクエストが同じキャッシュキーを共有しない場合、CloudFront は一意のキャッシュキーを持つすべてのリクエストをオリジンに転送します。

すべてのリクエストの折りたたみを防ぐ場合は、マネージドキャッシュポリシー `CachingDisabled` を使用できます (このポリシーはキャッシュも防ぎます)。詳細については、「マネージドキャッシュポリシーを使用する」を参照してください。

特定のオブジェクトでのリクエストの折りたたみを防ぐ場合は、キャッシュ動作の最小 TTL を 0 に設定し、さらに `Cache-Control: private`、`Cache-Control: no-store`、`Cache-Control:`

no-cache、Cache-Control: max-age=0、または Cache-Control: s-maxage=0 を送信するようにオリジンを設定できます。これらの設定に伴ってオリジンへの負荷が増え、CloudFront が最初のリクエストへのレスポンスを待機中に一時停止される同時リクエストのレイテンシーが高くなります。

> **Important**
>
> 現在、キャッシュポリシー、オリジンリクエストポリシー、またはレガシーキャッシュ設定で Cookie 転送を有効にした場合、CloudFront はリクエストの折りたたみをサポートしません。

CloudFront が Amazon S3 オリジンからのレスポンスを処理する方法

CloudFront が Amazon S3 オリジンからのレスポンスを処理する方法について説明します。

目次

取り消されたリクエスト

オブジェクトがエッジキャッシュになく、CloudFront がオブジェクトをオリジンから取得したものの、リクエストされたオブジェクトを配信する前にビューワーがセッションを終了すると (ブラウザを閉じるなど)、CloudFront はそのオブジェクトをエッジロケーションにキャッシュしません。

CloudFront が削除または更新する HTTP レスポンスヘッダー

CloudFront は、Amazon S3 オリジンからのレスポンスをビューワーに転送する前に、以下のヘッダーフィールドを削除または更新します。

- X-Amz-Id-2
- X-Amz-Request-Id

- `Set-Cookie` – Cookie を転送するように CloudFront を構成している場合、`Set-Cookie` ヘッダーフィールドがクライアントに転送されます。詳細については、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。
- `Trailer`
- `Transfer-Encoding` – Amazon S3 オリジンがこのヘッダーフィールドを返す場合、CloudFront は値を `chunked` に設定してからビューワーにレスポンスを返します。
- `Upgrade`
- `Via` – CloudFront は、ビューワーへの応答で値を次のように設定します。

  `Via: http-version alphanumeric-string.cloudfront.net (CloudFront)`

  例えば、値は次のようになります。

  `Via: 1.1 1026589cc7887e7a0dc7827b4example.cloudfront.net (CloudFront)`

キャッシュ可能なファイルの最大サイズ

CloudFront がキャッシュに保存するレスポンスボディの最大サイズは 50 GB です。これには、`Content-Length` ヘッダーの値を指定しないチャンク転送レスポンスが含まれます。

それぞれ 50 GB 以下のパートのオブジェクトをリクエストする範囲リクエストを使用して、このサイズを超えるオブジェクトをキャッシュするには、CloudFront を使用します。CloudFront がこれらのパートをキャッシュするのは、それぞれが 50 GB 以下であるためです。ビューワーによって、オブジェクトのすべてのパートを取得した後、元の大きなオブジェクトが再構築されます。詳しくは、「範囲リクエストを使用して大きなオブジェクトをキャッシュする」を参照してください。

リダイレクト

すべてのリクエストを別のホスト名にリダイレクトするように Amazon S3 バケットを構成できます。別のホスト名には、別の Amazon S3 バケットまたは HTTP サーバーを使用できます。すべてのリクエストをリダイレクトするようにバケットを構成しており、バケットが CloudFront ディストリビューションのオリジンの場合、ディストリビューションのドメイン名 (例: d111111abcdef8.cloudfront.net) またはディストリビューションに関連付けられた代替ドメイン名 (CNAME) (例: example.com) を使用してすべてのリクエストを CloudFront ディストリビューションにリダイレクトするようにバケットを構成することをお勧めします。このように構成しない場合、ビューワーリクエストは CloudFront をバイパスし、オブジェクトは新しいオリジンから直接提供されます。

> **Note**
>
> 代替ドメイン名にリクエストをリダイレクトする場合は、CNAME レコードを追加してドメインの DNS サービスを更新する必要もあります。詳細については、「代替ドメイン名 (CNAME) を追加することによって、カスタム URL を使用する」を参照してください。

すべてのリクエストをリダイレクトするようにバケットを構成した場合の動作を以下に示します。

1. ビューワー (例: ブラウザ) が CloudFront にオブジェクトを要求します。
2. CloudFront は、ディストリビューションのオリジンである Amazon S3 バケットにリクエストを転送します。
3. Amazon S3 は、HTTP ステータスコード 301 (Moved Permanently) と新しい場所を返します。
4. CloudFront は、リダイレクトのステータスコードと新しい場所をキャッシュし、ビューワーに値を返します。CloudFront がリダイレクトに従って新しい場所からオブジェクトを取得することはありません。
5. ビューワーはオブジェクトに対する別のリクエストを送信しますが、今回は、CloudFront から取得した新しい場所を指定します。

   - Amazon S3 バケットが、ディストリビューションのドメイン名または代替ドメイン名を使用してすべてのリクエストを CloudFront ディストリビューションにリダイレクトする場合、CloudFront は新しい場所にある Amazon S3 バケットまたは HTTP サーバーのオブジェクトをリクエストします。新しい場所からオブジェクトが返されると、CloudFront はオブジェクトをビューワーに返し、エッジロケーションにオブジェクトをキャッシュします。
   - Amazon S3 バケットがリクエストを別の場所にリダイレクトする場合、2 番目のリクエストは CloudFront をバイパスします。新しい場所にある Amazon S3 バケットまたは HTTP サーバーがオブジェクトをビューワーに直接返すので、オブジェクトは CloudFront エッジキャッシュに一切キャッシュされません。

カスタムオリジンの場合のリクエストおよびレスポンスの動作

カスタムオリジンの使用時に CloudFront がリクエストとレスポンスを処理する方法については、以下のセクションを参照してください。

トピック

- CloudFront がリクエストを処理してカスタムオリジンに転送する方法

- CloudFront がカスタムオリジンからのレスポンスを処理する方法

CloudFront がリクエストを処理してカスタムオリジンに転送する方法

CloudFront がビューワーのリクエストを処理してカスタムオリジンに転送する方法について説明します。

目次

認証

Authorization ヘッダーをオリジンに転送すると、次のタイプのリクエストに対してクライアント認証を要求するようにオリジンサーバーを設定できます。

- DELETE
- GET
- HEAD
- PATCH
- PUT
- POST

OPTIONS リクエストの場合、次の CloudFront 設定を使用した場合のみ、クライアント認証を設定できます。

- CloudFront は、Authorization ヘッダーをオリジンに転送するように設定されます。
- CloudFront は、OPTIONS リクエストへの応答をキャッシュしないように設定されます。

詳細については、「Authorization ヘッダーを転送するように CloudFront を設定する」を参照してください。

HTTP または HTTPS を使用して、リクエストをオリジンサーバーに転送できます。詳細については、「CloudFront で HTTPS を使用する」を参照してください。

キャッシュ期間および最小 TTL

CloudFront が別のリクエストをオリジンに転送するまでにオブジェクトを CloudFront キャッシュに保持する時間を制御するには、次の手順を実行します。

- Cache-Control または Expires ヘッダーフィールドを各オブジェクトに追加するようにオリジンを構成します。
- CloudFront キャッシュ動作で、最小 TTL の値を指定します。
- デフォルト値の 24 時間を使用します。

詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

クライアント IP アドレス

ビューワーがリクエストを CloudFront に送信し、X-Forwarded-For リクエストヘッダーを含めない場合、CloudFront は TCP 接続からビューワーの IP アドレスを取得して、IP アドレスが含まれた X-Forwarded-For ヘッダーを追加し、リクエストをオリジンに転送します。たとえば、CloudFront が TCP 接続から IP アドレス 192.0.2.2 を取得する場合、以下のヘッダーをオリジンに転送します。

```
X-Forwarded-For: 192.0.2.2
```

ビューワーがリクエストを CloudFront に転送して X-Forwarded-For リクエストヘッダーを含める場合、CloudFront はビューワーの IP アドレスを TCP 接続から取得してそれを X-Forwarded-For ヘッダーの末尾に追加し、リクエストをオリジンに転送します。たとえば、ビューワーのリクエストに X-Forwarded-For: 192.0.2.4,192.0.2.3 が含まれ、CloudFront が TCP 接続から IP アドレス 192.0.2.2 を取得する場合、以下のヘッダーをオリジンに転送します。

```
X-Forwarded-For: 192.0.2.4,192.0.2.3,192.0.2.2
```

ロードバランサー (Elastic Load Balancing を含む)、ウェブアプリケーションファイアウォール、リバースプロキシ、侵入防御システム、API Gateway などの一部のアプリケーションでは、リクエストを転送した CloudFront エッジサーバーの IP アドレスが X-Forwarded-For ヘッダーの末尾に付加されます。たとえば、CloudFront から ELB に転送するリクエストに X-Forwarded-For: 192.0.2.2 が含まれていて、CloudFront エッジサーバーの IP アドレスが 192.0.2.199 である場合、EC2 インスタンスで受け取るリクエストのヘッダーは次のようになります。

```
X-Forwarded-For: 192.0.2.2,192.0.2.199
```

> **Note**
>
> X-Forwarded-For ヘッダーには、IPv4 アドレス (192.0.2.44 など) および IPv6 アドレス (2001:0db8:85a3::8a2e:0370:7334 など) が含まれます。
> また、X-Forwarded-For ヘッダーは現在のサーバー (CloudFront) へのパス上のすべてのノードによって変更される可能性があることにも注意してください。詳細については、RFC 7239 のセクション 8.1 を参照してください。CloudFront エッジコンピューティング関数を使用してヘッダーを変更することもできます。

クライアント側の SSL 認証

CloudFront はクライアント側の SSL 証明書を使用したクライアント認証をサポートしていません。オリジンがクライアント側証明書をリクエストした場合、CloudFront はリクエストを削除します。

圧縮

詳しくは、「圧縮ファイルを供給する」を参照してください。

条件付きリクエスト

CloudFront は、エッジキャッシュで有効期限切れになっているオブジェクトに対するリクエストを受け取ると、リクエストをオリジンに転送し、オブジェクトの最新バージョンを取得するか、CloudFront エッジキャッシュに最新バージョンがすでに存在することをオリジンに確認します。通常、オリジンはオブジェクトを CloudFront に最後に送信するときに、`ETag` 値または `LastModified` 値、あるいはその両方の値をレスポンスに含めます。CloudFront は、CloudFront がオリジンに転送する新しいリクエストには、次のどちらかまたは両方を追加します。

- オブジェクトの有効期限切れバージョンの `If-Match` 値が含まれる `If-None-Match` または `ETag` ヘッダー。
- オブジェクトの有効期限切れバージョンの `If-Modified-Since` 値が含まれる `LastModified` ヘッダー。

オリジンは、この情報を使用して、オブジェクトが更新されているかどうかを判別します。つまり、オブジェクト全体を CloudFront に返すか、または HTTP 304 ステータスコード (変更なし) のみを返すかを判別します。

> **Note**
>
> `If-Modified-Since` と `If-None-Match` の条件付きリクエストは、CloudFront が Cookie (すべてまたはサブセット) を転送するように設定されている場合はサポートされません。詳細については、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。

Cookie

Cookie をオリジンに転送するように CloudFront を構成できます。詳細については、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。

クロスオリジンリソース共有 (CORS)

CloudFront で Cross-Origin Resource Sharing 設定を適用する場合は、`Origin` ヘッダーをオリジンに転送するように CloudFront を設定します。詳しくは、「リクエストヘッダーに基づいてコンテンツをキャッシュする」を参照してください。

暗号化

HTTPS を使用してリクエストを CloudFront に送信するようにビューワーに要求し、ビューワーが使用しているプロトコルを使用してカスタムオリジンにリクエストを転送するように CloudFront に要求することもできます。詳細については、次のディストリビューション設定を参照してください。

- ビューワープロトコルポリシー
- プロトコル (カスタムオリジンのみ)

CloudFront は、SSLv3、TLSv1.0、TLSv1.1 および TLSv1.2 プロトコルを使用して、HTTPS リクエストをオリジンサーバーに転送します。カスタムオリジンでは、オリジンと通信する際に CloudFront が使用する SSL プロトコルを選択できます。

- CloudFront コンソールを使用する場合は、[オリジン SSL プロトコル] チェックボックスを使用するプロトコルを選択します。詳細については、「ディストリビューションを作成する」を参照してください。
- CloudFront API を使用する場合は、`OriginSslProtocols` 要素を使用してプロトコルを指定します。詳細については、Amazon CloudFront API リファレンスの「OriginSslProtocols」および「DistributionConfig」を参照してください。

オリジンが Amazon S3 バケットの場合、CloudFront は常に TLSv1.2 を使用します。

> **Important**
>
> SSL と TLS のその他のバージョンはサポートされていません。

CloudFront での HTTPS の使用の詳細については、「CloudFront で HTTPS を使用する」を参照してください。ビューワーと CloudFront との間、および CloudFront とオリジンとの間の HTTPS 通信で CloudFront がサポートする暗号のリストについては、ビューワーと CloudFront との間でサポートされているプロトコルと暗号 を参照してください。

本文が含まれている GET リクエスト

ビューワー GET のリクエストの本文が含まれている場合、CloudFront はビューワーに HTTP ステータスコード 403 (禁止) を返します。

HTTP メソッド

サポートするすべての HTTP メソッドを処理するよう CloudFront を構成すると、CloudFront はビューワーからの以下のリクエストを受け入れてカスタムオリジンに転送します。

- DELETE
- GET
- HEAD
- OPTIONS
- PATCH
- POST
- PUT

CloudFront は、GET リクエストと HEAD リクエストへの応答を常にキャッシュします。OPTIONS リクエストへの応答をキャッシュするように CloudFront を設定することもできます。CloudFront はその他のメソッドを使用するリクエストへのレスポンスをキャッシュしません。

カスタムオリジンが上記のメソッドを処理するかどうかを構成する方法の詳細については、オリジンのドキュメントを参照してください。

> **Important**
>
> CloudFront がサポートするすべての HTTP メソッドを受け入れてオリジンに転送するように CloudFront を構成する場合、オリジンサーバーがすべてのメソッドを処理するように構成します。たとえば、POST を使用するために、上記のメソッドを受け入れて転送するように CloudFront を構成する場合は、DELETE リクエストを適切に処理するようオリジンサーバーを構成して、削除すべきでないリソースをビューワーが削除できないようにする必要があります。詳細については、HTTP サーバーのドキュメントを参照してください。

HTTP リクエストヘッダーと CloudFront の動作 (カスタムオリジンおよび Amazon S3 オリジン)

次の表は、カスタムオリジンと Amazon S3 オリジンの両方に転送できる HTTP リクエストヘッダーを示しています (例外も注記されています)。この表には、各ヘッダーについて以下に関する情報も含まれています。

- ヘッダーをオリジンに転送するように CloudFront を設定していない場合の CloudFront の動作。この場合、CloudFront はヘッダー値に基づいてオブジェクトをキャッシュします。
- そのヘッダーの値に基づいてオブジェクトをキャッシュするように CloudFront を設定できるかどうか。

  `Date` および `User-Agent` ヘッダーの値に基づいてオブジェクトをキャッシュするように CloudFront を設定できますが、これはお勧めできません。これらのヘッダーには可能な値が多数あり、その値に基づいてキャッシュすると、CloudFront がオリジンに転送するリクエストの数が大幅に増加します。

ヘッダー値に基づくキャッシュの詳細については、「リクエストヘッダーに基づいてコンテンツをキャッシュする」を参照してください。

ヘッダー	ヘッダー値に基づいてキャッシュするように CloudFront を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
他の定義されたヘッダー	レガシーキャッシュ設定 – CloudFront はヘッダーをオリジンに転送します。	はい
`Accept`	CloudFront はヘッダーを削除します。	はい
`Accept-Charset`	CloudFront はヘッダーを削除します。	はい
`Accept-Encoding`		はい

ヘッダー	ヘッダー値に基づいてキャッシュするように CloudFront を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
	値に gzip または br が含まれている場合、CloudFront は正規化された Accept-Encoding ヘッダーをオリジンに転送します。 詳細については、「圧縮のサポート」および「圧縮ファイルを供給する」を参照してください。	
Accept-Language	CloudFront はヘッダーを削除します。	はい
Authorization	• GET、HEAD の各リクエスト – CloudFront は、リクエストをオリジンに転送する前に Authorization ヘッダーフィールドを削除します。 • OPTIONS リクエスト – Authorization リクエストへの応答をキャッシュするように CloudFront を設定した場合、CloudFront は、リクエストをオリジンに転送する前に、OPTIONS ヘッダーフィールドを削除します。 OPTIONS リクエストへの応答をキャッシュするように CloudFront を設定しなかった場合、CloudFront は、Authorization ヘッダーフィールドをオリジンに転送します。 • DELETE、PATCH、POST、PUT の各リクエスト – CloudFront は、リクエストをオリジンに転送する前にヘッダーフィールドを削除しません。	はい

ヘッダー	ヘッダー値に基づいてキャッシュするように CloudFront を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
`Cache-Control`	CloudFront はヘッダーをオリジンに転送します。	いいえ
`CloudFront-Forwarded-Proto`	CloudFront は、リクエストをオリジンに転送する前にヘッダーを追加しません。 詳細については、「リクエストのプロトコルに基づいてキャッシュを設定する」を参照してください。	はい
`CloudFront-Is-Desktop-Viewer`	CloudFront は、リクエストをオリジンに転送する前にヘッダーを追加しません。 詳細については、「デバイスタイプに基づいてキャッシュを設定する」を参照してください。	はい
`CloudFront-Is-Mobile-Viewer`	CloudFront は、リクエストをオリジンに転送する前にヘッダーを追加しません。 詳細については、「デバイスタイプに基づいてキャッシュを設定する」を参照してください。	はい
`CloudFront-Is-Tablet-Viewer`	CloudFront は、リクエストをオリジンに転送する前にヘッダーを追加しません。 詳細については、「デバイスタイプに基づいてキャッシュを設定する」を参照してください。	はい
`CloudFront-Viewer-Country`	CloudFront は、リクエストをオリジンに転送する前にヘッダーを追加しません。	はい

ヘッダー	ヘッダー値に基づいてキャッシュするように CloudFront を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
Connection	CloudFront は、オリジンにリクエストを転送する前に、このヘッダーを Connection: Keep-Alive ヘッダーに置き換えます。	いいえ
Content-Length	CloudFront はヘッダーをオリジンに転送します。	いいえ
Content-MD5	CloudFront はヘッダーをオリジンに転送します。	はい
Content-Type	CloudFront はヘッダーをオリジンに転送します。	はい
Cookie	Cookie を転送するように CloudFront を設定すると、Cookie ヘッダーフィールドがオリジンに転送されます。そうでない場合、CloudFront は Cookie ヘッダーフィールドを削除します。詳細については、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。	いいえ
Date	CloudFront はヘッダーをオリジンに転送します。	はい、ただし推奨されません
Expect	CloudFront はヘッダーを削除します。	はい
From	CloudFront はヘッダーをオリジンに転送します。	はい

ヘッダー	ヘッダー値に基づいてキャッシュするように CloudFront を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
Host	CloudFront は、リクエストされたオブジェクトに関連付けられたオリジンのドメイン名に値を設定します。 Amazon S3 または MediaStore オリジンのホストヘッダーに基づいてキャッシュすることはできません。	はい (カスタム) いいえ (S3 および MediaStore)
If-Match	CloudFront はヘッダーをオリジンに転送します。	はい
If-Modified-Since	CloudFront はヘッダーをオリジンに転送します。	はい
If-None-Match	CloudFront はヘッダーをオリジンに転送します。	はい
If-Range	CloudFront はヘッダーをオリジンに転送します。	はい
If-Unmodified-Since	CloudFront はヘッダーをオリジンに転送します。	はい
Max-Forwards	CloudFront はヘッダーをオリジンに転送します。	いいえ
Origin	CloudFront はヘッダーをオリジンに転送します。	はい
Pragma	CloudFront はヘッダーをオリジンに転送します。	いいえ
Proxy-Authenticate	CloudFront はヘッダーを削除します。	いいえ

ヘッダー	ヘッダー値に基づいてキャッシュするように CloudFront を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
`Proxy-Authorization`	CloudFront はヘッダーを削除します。	いいえ
`Proxy-Connection`	CloudFront はヘッダーを削除します。	いいえ
`Range`	CloudFront はヘッダーをオリジンに転送します。詳細については、「CloudFront がオブジェクトの部分的リクエスト (レンジ GET) を処理する方法」を参照してください。	はい (デフォルト)
`Referer`	CloudFront はヘッダーを削除します。	はい
`Request-Range`	CloudFront はヘッダーをオリジンに転送します。	いいえ
`TE`	CloudFront はヘッダーを削除します。	いいえ
`Trailer`	CloudFront はヘッダーを削除します。	いいえ
`Transfer-Encoding`	CloudFront はヘッダーをオリジンに転送します。	いいえ
`Upgrade`	WebSocket 接続が確立されていない限り、CloudFront はヘッダーを削除します。	いいえ (WebSocket 接続の場合を除く)

ヘッダー	ヘッダー値に基づいてキャッシュするように CloudFront を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
User-Agent	CloudFront はこのヘッダーフィールドの値を Amazon CloudFront に置き換えます。ユーザーが使用しているデバイスに基づいて CloudFront でコンテンツをキャッシュする場合は、「デバイスタイプに基づいてキャッシュを設定する」を参照してください。	はい、ただし推奨されません
Via	CloudFront はヘッダーをオリジンに転送します。	はい
Warning	CloudFront はヘッダーをオリジンに転送します。	はい
X-Amz-Cf-Id	CloudFront は、リクエストをオリジンに転送する前に、ビューワーリクエストにヘッダーを追加します。ヘッダー値には、リクエストを一意に識別する暗号化された文字列が含められます。	いいえ
X-Edge-*	CloudFront はすべての X-Edge-* ヘッダーを削除します。	いいえ
X-Forwarded-For	CloudFront はヘッダーをオリジンに転送します。詳細については、「クライアント IP アドレス」を参照してください。	はい
X-Forwarded-Proto	CloudFront はヘッダーを削除します。	いいえ
X-HTTP-Method-Override	CloudFront はヘッダーを削除します。	はい

ヘッダー	ヘッダー値に基づいてキャッシュするように CloudFront を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
X-Real-IP	CloudFront はヘッダーを削除します。	いいえ

HTTP バージョン

CloudFront は HTTP/1.1 を使用してカスタムオリジンにリクエストを転送します。

リクエストの最大長と URL の最大長

パス、クエリ文字列 (ある場合)、ヘッダーを含め、リクエストの最大長は 20480 バイトです。

CloudFront はリクエストから URL を構築します。この URL の最大長は 8192 文字です。

リクエストまたは URL がこの最大制限を超えると、CloudFront は、リクエストヘッダーフィールドが長すぎることを示す HTTP ステータスコード 413 (Request Entity Too Large) をビューワーに返してから、ビューワーへの TCP 接続を終了します。

OCSP stapling

オブジェクトに対する HTTPS リクエストをビューワーが送信する際には、ドメインの SSL 証明書が無効になっていないことを CloudFront またはビューワーが認証機関 (CA) に対して確認する必要があります。OCSP Stapling を使用すると、CloudFront で証明書を検証して CA からの応答をキャッシュできるため、クライアントが直接 CA に対して証明書を検証する必要がなくなり、証明書の検証速度が向上します。

同一ドメイン内のオブジェクトに対する多数の HTTPS リクエストを CloudFront が受信した場合は、OCSP Stapling によるパフォーマンス向上がさらに顕著になります。CloudFront エッジロケーション内の各サーバーは、別々の検証リクエストを送信する必要があります。同一ドメインに対する多数の HTTPS リクエストを CloudFront が受信するとすぐに、エッジロケーション内のすべてのサーバーが、SSL ハンドシェイクでパケットに「ステープリング」できるという CA からの応答を受信します。証明書が有効であることをビューワーが確認すると、CloudFront はリクエストされたオブジェクトを提供できます。CloudFront エッジロケーション内でディストリビューションが十分なトラフィックを確保できない場合、新しいリクエストは、CA に対して証明書がまだ検証されてい

ないサーバーに誘導される可能性が高くなります。この場合は、ビューワーが検証ステップを別途実行し、CloudFront サーバーがオブジェクトを提供します。この CloudFront サーバーも CA に検証リクエストを送信するため、同じドメイン名が含まれるリクエストを次に受信したときには、CA からの検証応答が既に存在しているということになります。

持続的接続

CloudFront がオリジンからレスポンスを取得すると、その期間中に別のリクエストが届くのに備え、数秒間、接続を維持しようとします。持続的接続を維持すると、TCP 接続の再構築に必要な時間と後続のリクエストに対する別の TLS ハンドシェイクの実行に必要な時間を節約できます。

持続的接続の期間を設定する方法など、詳細については、「キープアライブタイムアウト (カスタムオリジンのみ)」セクションの「ディストリビューション設定リファレンス」を参照してください。

プロトコル

CloudFront は、以下の項目に基づいて、HTTP または HTTPS リクエストをオリジンサーバーに転送します。

- ビューワーが CloudFront に送信したリクエストのプロトコル (HTTP または HTTPS)。
- CloudFront コンソールの [オリジンプロトコルポリシー] フィールドの値。または、CloudFront API を使用する場合は、`OriginProtocolPolicy` 複合型の `DistributionConfig` エレメントの値。CloudFront コンソールで使用できるオプションは、[HTTP のみ]、[HTTPS のみ]、および [一致ビューワー] です。

[HTTP のみ] または [HTTPS のみ] を指定すると、CloudFront では、ビューワーリクエストのプロトコルに関係なく、指定されたプロトコルのみを使用してリクエストをオリジンサーバーに転送します。

[一致ビューワー] を指定した場合、CloudFront はビューワーリクエストのプロトコルを使用してリクエストをオリジンサーバーに転送します。ビューワーが HTTP と HTTPS の両方のプロトコルを使用してリクエストを行った場合も、CloudFront がオブジェクトをキャッシュするのは 1 回だけです。

> **Important**
>
> CloudFront が HTTPS プロトコルを使用してリクエストをオリジンに転送し、オリジンサーバーから無効な証明書または自己署名証明書が返された場合、CloudFront は TCP 接続を中断します。

CloudFront コンソールを使用してディストリビューションを更新する方法については、「ディストリビューションを更新する」を参照してください。CloudFront API を使用してディストリビューションを更新する方法については、Amazon CloudFront API リファレンスの「UpdateDistribution」を参照してくださいCloudFront。

クエリ文字列

CloudFront でクエリ文字列パラメータをオリジンに転送するかどうかを構成できます。詳細については、「クエリ文字列パラメータに基づいてコンテンツをキャッシュする」を参照してください。

オリジン接続のタイムアウトと試行

オリジン接続タイムアウトは、オリジンへの接続を確立しようとしたときに CloudFront が待機する秒数です。

オリジン接続試行は、CloudFront がオリジンへの接続を試行する回数です。

これらの設定により、セカンダリオリジンにフェイルオーバーするか (オリジングループの場合)、ビューワーにエラーレスポンスを返す前に、CloudFront がオリジンへの接続を試行する時間が決まります。デフォルトでは、CloudFront はセカンダリオリジンへの接続を試行したり、エラーレスポンスを返したりする前に 30 秒 (それぞれ 10 秒間の試行が 3 回) 待機します。接続タイムアウトを短くするか、試行回数を減らすか、その両方を行うことで、この時間を短縮できます。

詳細については、「オリジンのタイムアウトと試行を制御する」を参照してください。

オリジン応答タイムアウト

オリジン応答タイムアウト (オリジンの読み取りタイムアウトまたはオリジンリクエストタイムアウトとも呼ばれます) は、次の両方に適用されます。

- CloudFront がリクエストをオリジンに転送してからレスポンスを受け取るまでの待機時間 (秒)。
- CloudFront がオリジンからレスポンスのパケットを受け取ってから次のパケットを受け取るまでの待機時間 (秒)。

CloudFront の動作は、ビューワーリクエストの HTTP メソッドによって決まります。

- `GET` および `HEAD` リクエスト – 応答タイムアウトの期間内にオリジンが応答しない場合、または応答を停止した場合、CloudFront は接続を中断します。指定されたオリジン接続の試行回数が 1 回を超える場合、CloudFront は完全な応答の取得を再試行します。オリジン接続の試行回数設定の値で決められているように、CloudFront は最大 3 回試行します。最後の試行でもオリジンが応

答しない場合、CloudFront は同じオリジンのコンテンツに対する別のリクエストを受け取るまで接続を試みません。

- DELETE、OPTIONS、PATCH、PUT、POST リクエスト – オリジンが 30 秒以内に応答しない場合、CloudFront は接続を中断し、オリジンへの接続を再試行しません。クライアントは、必要に応じてリクエストを再送信できます。

オリジン応答タイムアウトを設定する方法など、詳細については、「応答タイムアウト (カスタムオリジンのみ)」を参照してください。

同一オブジェクトへの同時リクエスト (リクエストを折りたたむ)

CloudFront エッジロケーションがオブジェクトのリクエストを受け取り、オブジェクトが現在キャッシュにないか、有効期限が切れている場合、CloudFront はすぐにオリジンにリクエストを送信します。ただし、同一オブジェクトへの同時リクエストがある場合 (同じキャッシュキーを使用)、CloudFront が最初のリクエストへのレスポンスを受信する前に、同一オブジェクト (同じキャッシュキー) への追加のリクエストがエッジロケーションに届く場合、CloudFront は一時停止してから、追加のリクエストをオリジンに転送します。この短い一時停止により、オリジンでの負荷を減らすことができます。CloudFront は、一時停止中に受け取ったすべてのリクエストに、元のリクエストからのレスポンスを送信します。これはリクエストの折りたたみと呼ばれます。CloudFront ログでは、最初のリクエストは `x-edge-result-type` フィールドで `Miss` と識別され、折りたたまれたリクエストは `Hit` とし識別されます。CloudFront ログの詳細については、「the section called “CloudFront とエッジ関数のログ記録”」を参照してください。

CloudFront は、キャッシュキーを共有するリクエストのみを折りたたみます。リクエストヘッダーまたはクッキーまたはクエリ文字列に基づいてキャッシュするように CloudFront を設定した場合など、追加のリクエストが同じキャッシュキーを共有しない場合、CloudFront は一意のキャッシュキーを持つすべてのリクエストをオリジンに転送します。

すべてのリクエストの折りたたみを防ぐ場合は、マネージドキャッシュポリシー `CachingDisabled` を使用できます (このポリシーはキャッシュも防ぎます)。詳細については、「マネージドキャッシュポリシーを使用する」を参照してください。

特定のオブジェクトへのリクエストの折りたたみを防ぐ場合は、キャッシュ動作の最小 TTL を 0 に設定し、さらに `Cache-Control: private`、`Cache-Control: no-store`、`Cache-Control: no-cache`、`Cache-Control: max-age=0`、または `Cache-Control: s-maxage=0` を送信するようにオリジンを設定できます。これらの設定に伴ってオリジンへの負荷が増え、CloudFront が最初のリクエストへのレスポンスを待機中に一時停止される同時リクエストのレイテンシーが高くなります。

> **Important**
>
> 現在、キャッシュポリシー、オリジンリクエストポリシー、またはレガシーキャッシュ設定で Cookie 転送を有効にした場合、CloudFront はリクエストの折りたたみをサポートしません。

User-Agent ヘッダー

ユーザーがコンテンツの表示に使用しているデバイスに基づいて、オブジェクトの異なるバージョンを CloudFront でキャッシュするには、次の 1 つ以上のヘッダーをカスタムオリジンに転送するように CloudFront を設定することをお勧めします。

- `CloudFront-Is-Desktop-Viewer`
- `CloudFront-Is-Mobile-Viewer`
- `CloudFront-Is-SmartTV-Viewer`
- `CloudFront-Is-Tablet-Viewer`

CloudFront は、`User-Agent` ヘッダーの値に基づいて、これらのヘッダーの値を `true` または `false` に設定した後、リクエストをオリジンに転送します。デバイスが複数のカテゴリに属する場合は、複数の値が `true` になることがあります。たとえば、一部のタブレットデバイスに対して、CloudFront が `CloudFront-Is-Mobile-Viewer` と `CloudFront-Is-Tablet-Viewer` の両方を `true` に設定する場合があります。リクエストヘッダーに基づいてキャッシュするように CloudFront を設定する方法の詳細については、「リクエストヘッダーに基づいてコンテンツをキャッシュする」を参照してください。

`User-Agent` ヘッダーの値に基づいてオブジェクトをキャッシュするように CloudFront を設定できますが、これはお勧めできません。`User-Agent` ヘッダーには可能な値が多数あり、その値に基づいてキャッシュすると、CloudFront がオリジンに転送するリクエストの数が大幅に増加します。

CloudFront が `User-Agent` ヘッダーの値に基づいてオブジェクトをキャッシュするように設定しない場合、CloudFront は以下の値を指定した `User-Agent` ヘッダーを追加して、リクエストをオリジンに転送します。

```
User-Agent = Amazon CloudFront
```

CloudFront は、ビューワーからのリクエストに User-Agent ヘッダーが含まれているかどうかに関係なく、このヘッダーを追加します。ビューワーからのリクエストに User-Agent ヘッダーが含まれる場合、CloudFront はそのヘッダーを削除します。

CloudFront がカスタムオリジンからのレスポンスを処理する方法

CloudFront がカスタムオリジンからのレスポンスを処理する方法について説明します。

目次

100 Continue 件のレスポンス

オリジンは複数の 100-continue レスポンスを CloudFront に送信することはできません。最初の 100-continue レスポンスの後で、CloudFront は HTTP 200 OK レスポンスを予期します。オリジンが最初のレスポンスの後に別の 100-continue レスポンスを送信すると、CloudFront はエラーを返します。

キャッシュ

- オリジンサーバーが Date および Last-Modified ヘッダーフィールドに有効かつ正確な値を設定していることを確認します。
- 通常、CloudFront はオリジンからのレスポンスの Cache-Control: no-cache ヘッダーを優先します。例外については、「同一オブジェクトへの同時リクエスト (リクエストを折りたたむ)」を参照してください。

取り消されたリクエスト

オブジェクトがエッジキャッシュになく、CloudFront がオブジェクトをオリジンから取得したものの、リクエストされたオブジェクトを配信する前にビューワーがセッションを終了すると (ブラウザを閉じるなど)、CloudFront はそのオブジェクトをエッジロケーションにキャッシュしません。

コンテンツネゴシエーション

オリジンが応答で `Vary:*` を返し、対応するキャッシュ動作の [最小 TTL] の値が [0] の場合、CloudFront はオブジェクトをキャッシュしますが、そのオブジェクトの後続のすべてのリクエストをオリジンに転送して、キャッシュにオブジェクトの最新バージョンが含まれていることを確認します。CloudFront には、`If-None-Match` や `If-Modified-Since` などの条件付きヘッダーは含まれません。その結果、オリジンはすべてのリクエストに応じて CloudFront にオブジェクトを返します。

オリジンが応答で `Vary:*` を返し、対応するキャッシュ動作の [最小 TTL] の値が別の値になっている場合、CloudFront は「`Vary`」に記述されている方法で CloudFront が削除または置き換える HTTP レスポンスヘッダー ヘッダーを処理します。

cookie

キャッシュ動作の Cookie を有効にしており、オリジンが Cookie とオブジェクトを返す場合、CloudFront はオブジェクトと Cookie の両方をキャッシュします。これにより、オブジェクトのキャッシュ可能性が低下します。詳細については、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。

中断された TCP 接続

オリジンがオブジェクトを CloudFront に返している間に CloudFront とオリジン間の TCP 接続が中断した場合、CloudFront の動作は、オリジンが `Content-Length` ヘッダーをレスポンスに含めたかどうかによって異なります。

- Content-Length ヘッダーあり – CloudFront は、オブジェクトをオリジンから取得すると、ビューワーにオブジェクトを返します。ただし、`Content-Length` ヘッダーの値がオブジェクトのサイズに一致しない場合、CloudFront はオブジェクトをキャッシュしません。
- Transfer-Encoding: Chunked – CloudFront は、オブジェクトをオリジンから取得すると、ビューワーにオブジェクトを返します。ただし、チャンクレスポンスが完了していない場合、CloudFront はオブジェクトをキャッシュしません。

- Content-Length ヘッダーなし – CloudFront はオブジェクトをビューワーに返して、オブジェクトをキャッシュしますが、オブジェクトが完全でない場合があります。Content-Length ヘッダーがない場合、CloudFront は、TCP 接続が誤って中断されたか、または故意に中断されたかを判断できません。

Content-Length ヘッダーを追加して、CloudFront が不完全なオブジェクトをキャッシュしないように HTTP サーバーを設定することをお勧めします。

CloudFront が削除または置き換える HTTP レスポンスヘッダー

CloudFront は、オリジンからのレスポンスをビューワーに転送する前に、以下のヘッダーフィールドを削除または更新します。

- Set-Cookie – Cookie を転送するように CloudFront を構成している場合、Set-Cookie ヘッダーフィールドがクライアントに転送されます。詳細については、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。
- Trailer
- Transfer-Encoding – オリジンがこのヘッダーフィールドを返す場合、CloudFront は値を chunked に設定してビューワーにレスポンスを返します。
- Upgrade
- Vary – 次の点に注意してください。
  - デバイス固有のヘッダーのいずれかをオリジン (CloudFront-Is-Desktop-Viewer、CloudFront-Is-Mobile-Viewer、CloudFront-Is-SmartTV-Viewer、CloudFront-Is-Tablet-Viewer) に転送するように CloudFront を設定しており、オリジンが Vary:User-Agent を CloudFront に返すように設定している場合、CloudFront は Vary:User-Agent をビューワーに返します。詳細については、「デバイスタイプに基づいてキャッシュを設定する」を参照してください。
  - Accept-Encoding ヘッダーに、Cookie または Vary のいずれかを含めるよう設定した場合、CloudFront はビューワーへの応答にその値を含めます。
  - オリジンにヘッダーを転送するように CloudFront を設定し、Vary ヘッダー (例えば Vary:Accept-Charset,Accept-Language) の CloudFront にヘッダー名を返すようにオリジンを設定した場合、CloudFront はこれらの値を持つ Vary ヘッダーをビューワーに返します。
  - CloudFront が * ヘッダーの Vary 値を処理する詳細については、「コンテンツネゴシエーション」を参照してください。

  - Vary ヘッダーで他の値を含めるようにオリジンを設定している場合、CloudFront はその値を削除してからビューワーに応答を返します。
- Via – CloudFront は、ビューワーへの応答で値を次のように設定します。

  ```
  Via: http-version alphanumeric-string.cloudfront.net (CloudFront)
  ```

  例えば、値は次のようになります。

  ```
  Via: 1.1 1026589cc7887e7a0dc7827b4example.cloudfront.net (CloudFront)
  ```

キャッシュ可能なファイルの最大サイズ

CloudFront がキャッシュに保存するレスポンスボディの最大サイズは 50 GB です。これには、Content-Length ヘッダーの値を指定しないチャンク転送レスポンスが含まれます。

それぞれ 50 GB 以下のパートのオブジェクトをリクエストする範囲リクエストを使用して、このサイズを超えるオブジェクトをキャッシュするには、CloudFront を使用します。CloudFront がこれらのパートをキャッシュするのは、それぞれが 50 GB 以下であるためです。ビューワーによって、オブジェクトのすべてのパートを取得した後、元の大きなオブジェクトが再構築されます。詳しくは、「範囲リクエストを使用して大きなオブジェクトをキャッシュする」を参照してください。

使用できないオリジン

オリジンサーバーが使用できないときに、CloudFront がエッジキャッシュに存在するオブジェクトのリクエストを受け取り、そのオブジェクトが (たとえば、Cache-Control max-age ディレクティブに指定された期間が経過しているために) 有効期限切れになっている場合、CloudFront は有効期限切れバージョンのオブジェクトを提供するか、またはカスタムエラーページを提供します。カスタムエラーページを設定したときの CloudFront の動作の詳細については、「カスタムエラーページが設定されている場合に CloudFront がエラーを処理する方法」を参照してください。

場合によって、要求頻度の低いオブジェクトは削除されてエッジキャッシュで使用できなくなることがあります。CloudFront は、削除されたオブジェクトを提供することはできません。

リダイレクト

オリジンサーバーでオブジェクトの場所を変更した場合、リクエストを新しい場所にリダイレクトするようにウェブサーバーを構成できます。リダイレクトが構成された後、ビューワーがオブジェクトのリクエストを最初に送信したときに、CloudFront はリクエストをオリジンに送信し、オリジンは

リダイレクトで応答します (例: `302 Moved Temporarily`)。CloudFront はリダイレクトをキャッシュし、ビューワーにリダイレクトを返します。CloudFront はリダイレクトには従いません。

リクエストを以下のどちらかの場所にリダイレクトするようにウェブサーバーを構成できます。

- オリジンサーバーのオブジェクトの新しい URL。ビューワーが新しい URL へのリダイレクトに従う場合、ビューワーは CloudFront をバイパスし、オリジンに直接アクセスします。つまり、オリジンにあるオブジェクトの新しい URL にリクエストをリダイレクトしないことをお勧めします。
- オブジェクトの新しい CloudFront URL。新しい CloudFront URL を含むリクエストがビューワーから送信されると、CloudFront は、オリジンの新しい場所からオブジェクトを取得し、エッジロケーションにキャッシュした後、ビューワーにオブジェクトを返します。オブジェクトに対する以降のリクエストはエッジロケーションによって処理されます。これにより、オリジンのオブジェクトを要求するビューワーに関連するレイテンシーと負荷が回避されます。ただし、オブジェクトに対する新しいリクエストごとに、CloudFront への 2 つのリクエストに対する料金がかかります。

`Transfer-Encoding` ヘッダー

CloudFront は、chunked ヘッダーの `Transfer-Encoding` 値のみをサポートします。オリジンが `Transfer-Encoding: chunked` を返した場合、CloudFront は、エッジロケーションで受け取ったオブジェクトをクライアントに返し、そのオブジェクトをチャンク形式でキャッシュして後続のリクエストに備えます。

ビューワーが `Range GET` をリクエストし、オリジンが `Transfer-Encoding: chunked` を返す場合、CloudFront はリクエストされた範囲ではなくオブジェクト全体をビューワーに返します。

レスポンスのコンテンツ長を事前に決定できない場合は、チャンクエンコーディングを使用することをお勧めします。詳細については、「中断された TCP 接続」を参照してください。

オリジングループに対するリクエストとレスポンスの動作

オリジングループへのリクエストは、オリジングループとして設定されていないオリジンへのリクエストと同じように機能します。ただし、オリジンフェイルオーバーがある場合を除きます。他のオリジンと同様に、CloudFront がリクエストを受信し、コンテンツがすでにエッジロケーションにキャッシュされている場合、コンテンツはキャッシュからビューワーに配信されます。キャッシュミスがあり、オリジンがオリジングループである場合、ビューワーリクエストはオリジングループ内のプライマリオリジンに転送されます。

プライマリオリジンのリクエストとレスポンスの動作は、オリジングループにないオリジンの場合と同じです。詳細については、「Amazon S3 オリジンに対するリクエストとレスポンスの動作」および「カスタムオリジンの場合のリクエストおよびレスポンスの動作」を参照してください。

以下にプライマリオリジンが特定の HTTP ステータスコードを返す場合のオリジンフェイルオーバーの動作を示します。

- HTTP 2xx ステータスコード (成功): CloudFront は、ファイルをキャッシュしてビューワーに返します。
- HTTP 3xx ステータスコード (リダイレクト): CloudFront は、ステータスコードをビューワーに返します。
- HTTP 4xx または 5xx ステータスコード (クライアント/サーバーエラー): 返されたステータスコードがフェイルオーバー用に設定されている場合、CloudFront はオリジングループのセカンダリオリジンに同じリクエストを送信します。
- HTTP 4xx または 5xx ステータスコード (クライアント/サーバーエラー): 返されたステータスコードがフェイルオーバー用に設定されていない場合、CloudFront はビューワーにエラーを返します。

CloudFront は、ビューワーリクエストの HTTP メソッドが GET、HEAD、または OPTIONS の場合にのみ、セカンダリオリジンにフェイルオーバーします。ビューワーが別の HTTP メソッド (たとえば POST や PUT など) を送信しても、CloudFront はフェイルオーバーしません。

CloudFront がセカンダリオリジンにリクエストを送信する場合、レスポンスの動作は、オリジングループにない CloudFront オリジンの場合と同じです。

オリジングループの詳細については、「CloudFront オリジンフェイルオーバーを使用して高可用性を最適化する」を参照してください。

オリジンリクエストにカスタムヘッダーを追加する

オリジンに送信するリクエストにカスタムヘッダーを追加するように CloudFront を設定できます。カスタムヘッダーを使用すると、一般的なビューワーリクエストでは得られない情報をオリジンから送信および収集できます。ヘッダーは、オリジンごとにカスタマイズすることもできます。CloudFront は、カスタムオリジンと Amazon S3 オリジンの両方でカスタムヘッダーをサポートしています。

目次

ユースケース

カスタムヘッダーは、以下の例に示すように使用できます。

CloudFront からのリクエストの識別

オリジンが CloudFront から受け取るリクエストを識別できます。これは、ユーザーが CloudFront をバイパスするかどうかを知りたい場合、または複数の CDN を使用してどのリクエストが CDN から来ているのかを知りたい場合に役立つ可能性があります。

> **Note**
>
> Amazon S3 オリジンを使用して、Amazon S3 サーバーアクセスログを有効にした場合、ログにヘッダー情報は含まれません。

特定のディストリビューションから送信されたリクエストの判断

同じオリジンを使用するように複数の CloudFront ディストリビューションを設定する場合は、ディストリビューションごとに異なるカスタムヘッダーを追加できます。その後、オリジンからのログを使用して、どのリクエストがどの CloudFront ディストリビューションから来たのかを判断できます。

Cross-Origin Resource Sharing (CORS) の有効化

ビューワーの一部が Cross-Origin Resource Sharing (CORS) をサポートしていない場合は、オリジンに送信されるリクエストに `Origin` ヘッダーを常に追加するように CloudFront を設定できます。次に、リクエストごとに `Access-Control-Allow-Origin` ヘッダーを返すようにオリジンを設定できます。CORS 設定を適用するように CloudFront を設定する必要もあります。

コンテンツへのアクセス制御

カスタムヘッダーを使用して、コンテンツへのアクセスを制御できます。CloudFront によって追加されたカスタムヘッダーが含まれている場合にのみリクエストに応答するようオリジンを設定することで、ユーザーが CloudFront をバイパスして、オリジンで直接コンテンツにアクセスする

ことを防ぐことができます。詳細については、「カスタムオリジンのファイルへのアクセスを制限する」を参照してください。

オリジンリクエストにカスタムヘッダーを追加するように CloudFront を設定する

オリジンに送信されるリクエストにカスタムヘッダーを追加するようディストリビューションを設定するには、次のいずれかの方法を使用してオリジン設定を更新します。

- CloudFront コンソール – ディストリビューションを作成または更新する場合、[カスタムヘッダーを追加] 設定でヘッダー名と値を指定します。詳細については、「カスタムヘッダーを追加する」を参照してください。
- CloudFront API – カスタムヘッダーを追加するオリジンごとに、`Origin` 内の `CustomHeaders` フィールドでヘッダー名と値を指定します。詳細については、「Amazon CloudFront API リファレンス」の「CreateDistribution」または「UpdateDistribution」を参照してください。

指定するヘッダーの名前と値がまだビューワーのリクエストに存在しない場合、CloudFront がそれらをオリジンリクエストに追加します。ヘッダーが存在する場合、CloudFront はリクエストをオリジンに転送する前にヘッダー値を上書きします。

オリジンカスタムヘッダーに適用されるクォータについては、「ヘッダーのクォータ」を参照してください。

CloudFront でオリジンリクエストに追加できないカスタムヘッダー

オリジンに送信されるリクエストに以下のヘッダーを追加するように CloudFront を設定することはできません。

- `Cache-Control`
- `Connection`
- `Content-Length`
- `Cookie`
- `Host`
- `If-Match`
- `If-Modified-Since`
- `If-None-Match`

- If-Range
- If-Unmodified-Since
- Max-Forwards
- Pragma
- Proxy-Authorization
- Proxy-Connection
- Range
- Request-Range
- TE
- Trailer
- Transfer-Encoding
- Upgrade
- Via
- X-Amz- で始まるヘッダー
- X-Edge- で始まるヘッダー
- X-Real-Ip

Authorization ヘッダーを転送するように CloudFront を設定する

ビューワーリクエストをオリジンに転送する際、CloudFront はデフォルトで一部のビューワーヘッダー (Authorization ヘッダーを含む) を削除します。オリジンがオリジンリクエストの Authorization ヘッダーを常に受け取るようにするには、次のオプションがあります。

- キャッシュポリシーを使用して、Authorization ヘッダーをキャッシュキーに追加します。キャッシュキー内のすべてのヘッダーは、オリジンリクエストに自動的に含まれます。詳細については、「ポリシーを使用してキャッシュキーを制御する」を参照してください。
- オリジンリクエストポリシーを使用し、すべてのビューワーヘッダーをオリジンに転送します。オリジンリクエストポリシーで Authorization ヘッダーを個別に転送することはできません。ただし、ビューワーヘッダーをすべて転送する場合には、Authorization ヘッダーが CloudFront によりビューワーリクエストに含められます。CloudFront では、このユースケースのために、Managed-AllViewer と呼ばれるマネージド型のオリジンリクエストポリシーが提供されています。詳細については、「マネージドオリジンリクエストポリシーを使用する」を参照してください。

CloudFront がオブジェクトの部分的リクエスト (レンジ GET) を処理する方法

大きなオブジェクトの場合、ビューワー (ウェブブラウザまたはその他のクライアント) は、複数のGET リクエストを実行し、Range リクエストヘッダーを使用して、小さいパートでオブジェクトをダウンロードできます。Range GET リクエストとも呼ばれるこのバイト範囲のリクエストでは、部分的なダウンロードの効率、および部分的に失敗した転送からの回復の効率が向上します。

CloudFront は Range GET リクエストを受け取ると、そのリクエストを受け取ったエッジロケーションのキャッシュをチェックします。そのエッジロケーションのキャッシュに、オブジェクト全体またはオブジェクトの要求されたパートがすでに含まれている場合、CloudFront は要求された範囲をキャッシュから直ちに提供します。

要求された範囲がキャッシュに含まれていない場合、CloudFront はリクエストをオリジンに転送します。(パフォーマンスを最適化するために、CloudFront は、Range GET でクライアントから要求された範囲よりも大きい範囲を要求する場合があります)。次に実行される処理は、オリジンが Range GET リクエストをサポートするかどうかによって異なります。

- オリジンが **Range GET** リクエストをサポートする場合: 要求した範囲が返されます。CloudFront は要求された範囲を提供し、今後のリクエストのためにその範囲をキャッシュします。(Amazon S3 も多くの HTTP サーバーも Range GET リクエストをサポートしています)。
- オリジンが **Range GET** リクエストをサポートしない場合: オブジェクト全体が返されます。CloudFront は、オブジェクト全体を送信して現在のリクエストを処理し、今後のリクエストのためにオブジェクトをキャッシュします。CloudFront はオブジェクト全体をエッジキャッシュにキャッシュした後、要求された範囲を提供することで新しい Range GET リクエストに応答します。

どちらの場合も、CloudFront は、オリジンから 1 バイト目が到着した直後に、要求された範囲またはオブジェクトをエンドユーザーに供給します。

> **Note**
>
> ビューワーが Range GET をリクエストし、オリジンが `Transfer-Encoding: chunked` を返す場合、CloudFront はリクエストされた範囲ではなくオブジェクト全体をビューワーに返します。

通常、CloudFront は、Range ヘッダーの RFC 仕様に準拠します。ただし、Range ヘッダーが以下の要件に準拠しない場合、CloudFront は、指定された範囲とステータスコード 200 を返す代わりに、完全なオブジェクトと HTTP ステータスコード 206 を返します。

- 範囲は昇順に指定されている必要があります。たとえば、100-200,300-400 は有効で、300-400,100-200 は無効です。
- 範囲は重複できません。たとえば、100-200,150-250 は無効です。
- すべての範囲指定が有効である必要があります。たとえば、範囲の一部に負の値を指定することはできません。

Rangeリクエストヘッダーの詳細については、RFC 7233 の「範囲リクエスト」または MDN Web ドキュメントの「範囲」を参照してください。

範囲リクエストを使用して大きなオブジェクトをキャッシュする

キャッシュが有効になっている場合、CloudFront では 50 GB を超えるオブジェクトは取得またはキャッシュされません。オリジンによって、オブジェクトがこのサイズを超えていることが示される場合 (Content-Lengthレスポンスヘッダーで)、CloudFront はオリジンへの接続を閉じて、ビューワーにエラーを返します。(キャッシュを無効にすることで、CloudFront はオリジンからこのサイズを超えるオブジェクトを取得し、それをビューワーに転送できます。ただし、CloudFront でオブジェクトはキャッシュされません)。

ただし、範囲リクエストで CloudFront を使用してキャッシュ可能な最大ファイルサイズを超えるサイズのオブジェクトをキャッシュできます。

Example 例

1. 100 GB のオブジェクトを持つオリジンについて考えてみます。キャッシュが有効になっている場合、CloudFront では、オブジェクトはこの大きさで取得またはキャッシュされません。ただし、ビューワーは複数の範囲リクエストを送信することにより、それぞれサイズが 50 GB 未満のパートを使用して、このパートのオブジェクトを取得できます。
2. ビューワーは、オブジェクトを 20 GB のパートに分けてリクエストする場合、ヘッダー Range: bytes=0-21474836480 を使用したリクエストを送信して最初のパートを取得し、次にヘッダー Range: bytes=21474836481-42949672960 を使用した別のリクエストで次のパートを取得できます。以下同様です。
3. ビューワーがすべてのパートを受信すると、それらを組み合わせて元の 100 GB のオブジェクトを構築できます。

4. この場合、CloudFront はそれぞれ 20 GB のパートのオブジェクトをキャッシュし、キャッシュの同じパートの後続リクエストに対応できます。

CloudFront がオリジンからの HTTP 3xx ステータスコードを処理する方法

CloudFront が Amazon S3 バケットまたはカスタムオリジンサーバーのオブジェクトをリクエストすると、オリジンサーバーは HTTP 3xx ステータスコードを返すことがあります。これは、次のいずれかを示しています。

- オブジェクトの URL が変更された (たとえば、ステータスコード 301、302、307、308)
- 前回 CloudFront がリクエストしてからオブジェクトが変更されていない (ステータスコード 304)

CloudFront は、CloudFront ディストリビューションの設定とレスポンスのヘッダーに従って 3xx レスポンスをキャッシュします。CloudFront は、オリジンからの応答に `Cache-Control` ヘッダーを含めた場合のみ、307 応答と 308 応答をキャッシュします。詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

オリジンからリダイレクトステータスコード (301 や 307 など) が返された場合、CloudFront はリダイレクトに従いません。CloudFront は、301 または 307 レスポンスをビューワーに渡します。ビューワーは、新しいリクエストを送信することでリダイレクトに従うことができます。

CloudFront がオリジンからの HTTP 4xx および 5xx ステータスコードを処理する方法

CloudFront が Amazon S3 バケットまたはカスタムオリジンサーバーのオブジェクトをリクエストすると、オリジンサーバーは HTTP 4xx または 5xx ステータスコードを返すことがあります。このステータスコードは、エラーが発生したことを示します。CloudFront の動作は、以下の条件によって左右されます。

- カスタムエラーページを設定しているかどうか
- オリジンからのエラーレスポンスを CloudFront がキャッシュする期間 (エラーキャッシュの最小 TTL) を設定しているかどうか
- ステータスコード

- 5xx ステータスコードの場合、リクエストされたオブジェクトが現在 CloudFront エッジキャッシュにあるかどうか
- 一部の 4xx ステータスコードの場合、オリジンが `Cache-Control max-age` または `Cache-Control s-maxage` ヘッダーを返すかどうか

CloudFront は、`GET` リクエストと `HEAD` リクエストへの応答を常にキャッシュします。`OPTIONS` リクエストへの応答をキャッシュするように CloudFront を設定することもできます。CloudFront はその他のメソッドを使用するリクエストへのレスポンスをキャッシュしません。

オリジンが応答しない場合、オリジンへの CloudFront リクエストはタイムアウトし、オリジンからの HTTP 5xx エラーと見なされます。これは、オリジンからそのエラーが返されなくても同様です。そのシナリオでは、CloudFront はキャッシュされたコンテンツを引き続き提供します。詳細については、「使用できないオリジン」を参照してください。

ログ作成を有効にしている場合、CloudFront は、HTTP ステータスコードに関係なく結果をログに書き込みます。

CloudFront から返されるエラーメッセージに関連する機能とオプションの詳細については、以下を参照してください。

- CloudFront コンソールでのカスタムエラーページの設定の詳細については、「カスタムエラーページとエラーキャッシュ」を参照してください。
- CloudFront コンソールでのエラーキャッシュ最小 TTL の詳細については、「Error caching minimum TTL (seconds) (エラーキャッシュ最小 TTL (秒))」を参照してください。
- CloudFront がキャッシュする HTTP ステータスコードのリストについては、「CloudFront がキャッシュする HTTP 4xx および 5xx ステータスコード」を参照してください。

トピック

- カスタムエラーページが設定されている場合に CloudFront がエラーを処理する方法
- カスタムエラーページが設定されていない場合に CloudFront がエラーを処理する方法
- CloudFront がキャッシュする HTTP 4xx および 5xx ステータスコード

カスタムエラーページが設定されている場合に CloudFront がエラーを処理する方法

カスタムエラーページを設定している場合、CloudFront の動作は、リクエストされたオブジェクトがエッジキャッシュにあるかどうかによって異なります。

リクエストされたオブジェクトがエッジキャッシュにない場合

CloudFront は、以下のすべてが該当する場合に、オリジンからリクエストされたオブジェクトの取得を試行し続けます。

- ビューワーがオブジェクトを要求する
- オブジェクトがエッジキャッシュにない
- オリジンが HTTP 4xx または 5xx ステータスコードを返して、以下のいずれかに該当する:
  - オリジンがステータスコード 304 (変更なし) またはオブジェクトの更新バージョンを返す代わりに HTTP 5xx ステータスコードを返す
  - オリジンが、キャッシュコントロールヘッダーによって制限されておらず、以下のステータスコードのリストに含まれている HTTP 4xx ステータスコードを返す: CloudFront が常にキャッシュする HTTP 4xx および 5xx ステータスコード。
  - オリジンが `Cache-Control max-age` ヘッダーまたは `Cache-Control s-maxage` ヘッダーなしで HTTP 4xx ステータスコードを返す。ステータスコードは、次のステータスコードの一覧に含まれる: コントロール CloudFront が `Cache-Control` ヘッダーに基づいてキャッシュする HTTP 4xx ステータスコード。

CloudFront は以下の処理を行います。

1. ビューワーからリクエストを受け取った CloudFront エッジキャッシュで、CloudFront はディストリビューション設定を確認し、オリジンから返されたステータスコードに対応するカスタムエラーページのパスを取得します。
2. CloudFront は、カスタムエラーページのパスと一致するパスパターンを持つ、ディストリビューション内の最初のキャッシュ動作を検索します。
3. CloudFront エッジロケーションは、キャッシュ動作に指定されているオリジンに、カスタムエラーページのリクエストを送信します。
4. オリジンはカスタムエラーページをエッジロケーションに返します。
5. CloudFront は、リクエストを送信したビューワーにカスタムエラーページを返します。また、最大で次の値になるようにカスタムエラーページをキャッシュします。

- エラーキャッシュ最小 TTL で指定された時間の長さ (デフォルトでは 10 秒)
- `Cache-Control max-age` ヘッダー、または最初のリクエストがエラーを生成したときに発信元から返された `Cache-Control s-maxage` ヘッダーで指定された時間

6. キャッシュ時間 (ステップ 5 で決定されます) が経過すると、CloudFront はオリジンに別のリクエストを転送して、リクエストされたオブジェクトの取得を再試行します。CloudFront は、エラーキャッシュ最小 TTL に指定された間隔で再試行し続けます。

リクエストされたオブジェクトがエッジキャッシュにある場合

CloudFront は、以下のすべてに該当する場合に、現在エッジキャッシュに存在するオブジェクトを引き続き提供します。

- ビューワーがオブジェクトを要求する
- オブジェクトがエッジキャッシュに存在するが有効期限が切れている
- オリジンがステータスコード 304 (変更なし) またはオブジェクトの更新バージョンを返す代わりに HTTP 5xx ステータスコードを返す

CloudFront は以下の処理を行います。

1. オリジンが 5xx ステータスコードを返した場合、CloudFront はオブジェクトの有効期限が切れていても、そのオブジェクトを返します。エラーキャッシュ最小 TTL の期間、CloudFront は、エッジキャッシュからオブジェクトを提供することで、ビューワーリクエストに対して応答し続けます。

   オリジンが 4xx ステータスコードを返した場合、CloudFront はリクエストされたオブジェクトではなく、ステータスコードをビューワーに返します。
2. エラーキャッシュ最小 TTL が経過すると、CloudFront はオリジンに別のリクエストを転送して、リクエストされたオブジェクトの取得を再試行します。オブジェクトが頻繁にリクエストされない場合、CloudFront はそのオブジェクトをエッジキャッシュから削除することがありますが、オリジンサーバーは引き続き 5xx レスポンスを返します。オブジェクトが CloudFront エッジキャッシュに保持される期間については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

カスタムエラーページが設定されていない場合に CloudFront がエラーを処理する方法

カスタムエラーページを設定していない場合、CloudFront の動作は、リクエストされたオブジェクトがエッジキャッシュにあるかどうかによって異なります。

リクエストされたオブジェクトがエッジキャッシュにない場合

CloudFront は、以下のすべてが該当する場合に、オリジンからリクエストされたオブジェクトの取得を試行し続けます。

- ビューワーがオブジェクトを要求する
- オブジェクトがエッジキャッシュにない
- オリジンが HTTP 4xx または 5xx ステータスコードを返して、以下のいずれかに該当する:
  - オリジンがステータスコード 304 (変更なし) またはオブジェクトの更新バージョンを返す代わりに HTTP 5xx ステータスコードを返す
  - オリジンがキャッシュコントロールヘッダーによって制限されておらず、以下のステータスコードのリストに含まれている HTTP 4xx ステータスコードを返す: CloudFront が常にキャッシュする HTTP 4xx および 5xx ステータスコード。
  - オリジンが `Cache-Control max-age` ヘッダーまたは `Cache-Control s-maxage` ヘッダーなしで HTTP 4xx ステータスコードを返す。ステータスコードは、次のステータスコードの一覧に含まれる: コントロール CloudFront が `Cache-Control` ヘッダーに基づいてキャッシュする HTTP 4xx ステータスコード。

CloudFront は以下の処理を行います。

1. CloudFront は 4xx または 5xx のステータスコードをビューワーに返します。また、次の最大値のリクエストを受け取ったエッジキャッシュにステータスコードをキャッシュします。
   - エラーキャッシュ最小 TTL で指定された時間の長さ (デフォルトでは 10 秒)
   - `Cache-Control max-age` ヘッダー、または最初のリクエストがエラーを生成したときに発信元から返された `Cache-Control s-maxage` ヘッダーで指定された時間
2. キャッシュ時間の期間 (ステップ 1 で決定されます) では、CloudFront はキャッシュされた 4xx または 5xx のステータスコードを使用して、同じオブジェクトに対する後続のビューワーリクエストに応答します。

3. キャッシュ時間 (ステップ 1 で決定されます) が経過すると、CloudFront はオリジンに別のリクエストを転送して、リクエストされたオブジェクトの取得を再試行します。CloudFront は、エラーキャッシュ最小 TTL に指定された間隔で再試行し続けます。

リクエストされたオブジェクトがエッジキャッシュにある場合

CloudFront は、以下のすべてに該当する場合に、現在エッジキャッシュに存在するオブジェクトを引き続き提供します。

- ビューワーがオブジェクトを要求する
- オブジェクトがエッジキャッシュに存在するが有効期限が切れている
- オリジンがステータスコード 304 (変更なし) またはオブジェクトの更新バージョンを返す代わりに HTTP 5xx ステータスコードを返す

CloudFront は以下の処理を行います。

1. オリジンが 5xx のエラーコードを返した場合、CloudFront は、オブジェクトの有効期限が切れていても、そのオブジェクトを返します。エラーキャッシュ最小 TTL 時間 (デフォルトでは 10 秒) の間、CloudFront は、エッジキャッシュからオブジェクトを提供することで、ビューワーからのリクエストに応答し続けます。

   オリジンが 4xx ステータスコードを返した場合、CloudFront はリクエストされたオブジェクトではなく、ステータスコードをビューワーに返します。
2. エラーキャッシュ最小 TTL が経過すると、CloudFront はオリジンに別のリクエストを転送して、リクエストされたオブジェクトの取得を再試行します。オブジェクトが頻繁にリクエストされない場合、CloudFront はそのオブジェクトをエッジキャッシュから削除することがありますが、オリジンサーバーは引き続き 5xx レスポンスを返します。オブジェクトが CloudFront エッジキャッシュに保持される期間については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

CloudFront がキャッシュする HTTP 4xx および 5xx ステータスコード

CloudFront は、返された特定のステータスコードと、オリジンがレスポンスで特定のヘッダーを返すかどうかに応じて、オリジンから返された HTTP 4xx と 5xx ステータスコードをキャッシュします。

CloudFront が常にキャッシュする HTTP 4xx および 5xx ステータスコード

CloudFront は、オリジンから返される以下の HTTP 4xx および 5xx ステータスコードを常にキャッシュします。HTTP ステータスコードのカスタムエラーページを設定している場合、CloudFront はカスタムエラーページをキャッシュします。

404	Not Found
414	Request-URI Too Large
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Time-out

CloudFront が **`Cache-Control`** ヘッダーに基づいてキャッシュする HTTP 4xx ステータスコード

オリジンが `Cache-Control max-age` または `Cache-Control s-maxage` ヘッダーを返す場合、CloudFront は以下の HTTP 4 xx ステータスコードのみをキャッシュします。これらの HTTP ステータスコードの 1 つに対してカスタムエラーページを設定していて、オリジンから 1 つのキャッシュコントロールヘッダーが返された場合、CloudFront はカスタムエラーページをキャッシュします。

400	Bad Request
403	Forbidden

405	Method Not Allowed
412[1]	Precondition Failed
415[1]	Unsupported Media Type

[1]CloudFront は、これらの HTTP ステータスコードに対するカスタムエラーページの作成をサポートしていません。

カスタムエラーレスポンスを生成する

CloudFront から配信されているオブジェクトが何らかの理由で使用できなくなった場合、これを伝えるために、通常はウェブサーバーから、関連する HTTP ステータスコードが CloudFront に返されます。例えば、ビューワーが無効な URL をリクエストした場合、HTTP 404 (Not Found) ステータスコードがウェブサーバーから CloudFront に返され、さらに CloudFront からビューワーに返されます。このデフォルトのエラーレスポンスを使用する代わりに、カスタムレスポンスを作成して CloudFront からビューワーに返すことができます。

HTTP ステータスコードの代わりにカスタムエラーページを返すように CloudFront を構成していても、カスタムエラーページが利用できない場合には、CloudFront は、カスタムエラーページを持つオリジンから受信したステータスコードをビューワーに返します。たとえば、カスタムオリジンから 500 ステータスコードが返され、500 ステータスコードのためのカスタムエラーページを Amazon S3 バケットから取得するように CloudFront を構成してあるとします。しかし、誰かが間違えてカスタムエラーページを Amazon S3 バケットから削除してしまいました。この場合 CloudFront は、そのオブジェクトをリクエストしたビューワーに対して、HTTP 404 ステータスコード (Not found) を返します。

CloudFront がカスタムエラーページをビューワーに返したときに、リクエストしたオブジェクトの料金ではなく、カスタムエラーページの標準 CloudFront 料金を支払います。CloudFront の料金の詳細については、「Amazon CloudFront 料金表」を参照してください。

トピック

- エラーレスポンスの動作を設定する
- HTTP ステータスコード別のカスタムエラーページを作成する

- オブジェクトとカスタムエラーページを別々の場所に保存する
- CloudFront から返されるレスポンスコードを変更する
- CloudFront がエラーをキャッシュする時間を制御する

エラーレスポンスの動作を設定する

エラーが発生した場合に CloudFront がどのように対応するかを管理するための、いくつかのオプションがあります。カスタムエラーレスポンスの設定は、CloudFront コンソール、CloudFront API、または から行えますAWS CloudFormation これらの内どれにより設定を更新する場合でも、次に挙げるヒントと推奨事項を参考にしてください。

- カスタムエラーページは、CloudFront からのアクセスが可能な場所に保存します。これらのページの保存先は、Amazon S3 バケットにすることを推奨します。また、ウェブサイトやアプリケーションなど、他のコンテンツと同じ場所には保存しないようにしてください。カスタムエラーページが、ウェブサイトやアプリケーションと同じオリジンに保存されている場合、オリジンのサーバーが使用不能になり 5xx エラーが返信されるようになると、CloudFront は、その使用不能なオリジンからカスタムエラーページを取得できなくなります。詳細については、「」を参照してくださいオブジェクトとカスタムエラーページを別々の場所に保存する
- CloudFront にカスタムエラーページを取得するための権限があることを確認します。カスタムエラーページを Amazon S3 に保存している場合、そのページがパブリックにアクセスが可能であるか、CloudFront のオリジンアクセスコントロール (OAI) を設定する必要があります。カスタムエラーページをカスタムオリジンに格納する場合には、そのページはパブリックにアクセス可能である必要があります。
- (オプション) 必要に応じてオリジンを設定し、カスタムエラーページに `Cache-Control` または `Expires` ヘッダーを追加します。また、エラーキャッシュ最小 TTL 設定を使用して、CloudFront がカスタムエラーページをキャッシュに保持する時間を制御することもできます。詳細については、「CloudFront がエラーをキャッシュする時間を制御する」を参照してください。

カスタムエラーレスポンスを設定する

CloudFront コンソールからカスタムエラーレスポンスを設定するには、CloudFront ディストリビューションが必要です。コンソールからカスタムエラーレスポンスの構成を設定する際には、ディストリビューションが既に用意されている必要があります。ディストリビューションの作成方法については、「基本的な CloudFront ディストリビューションの開始方法」を参照してください。

Console

カスタムエラーレスポンスを設定するには (コンソール)

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home#distributionsの CloudFront コンソールで [ディストリビューション] ページを開きます。
2. ディストリビューションの一覧で、更新するディストリビューションを選択します。
3. [エラーページ] タブを開き、[カスタムエラーレスポンスの作成] をクリックします。
4. 適切な値を入力します。詳細については、「カスタムエラーページとエラーキャッシュ」を参照してください。
5. 必要な値を入力したら、[作成] をクリックします。

CloudFront API or AWS CloudFormation

CloudFront API または AWS CloudFormation でカスタムエラーレスポンスを設定するには、`CustomErrorResponse` タイプのディストリビューションを使用します。詳細については、以下を参照してください。

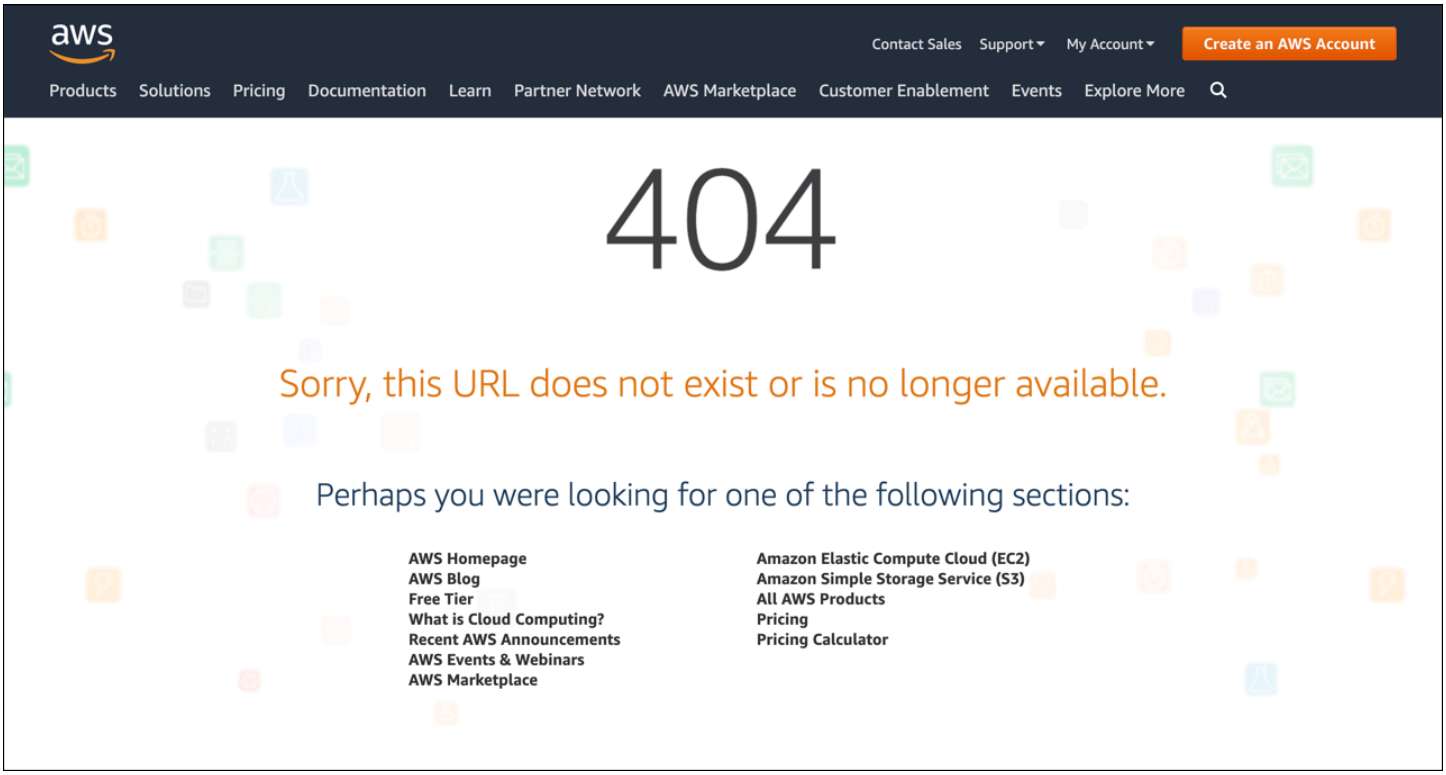
- AWS:: CloudFront:: ディストリビューション CustomErrorResponse (AWS CloudFormation ユーザーガイド)
- Amazon CloudFront API リファレンスの「CustomErrorResponse」

HTTP ステータスコード別のカスタムエラーページを作成する

デフォルト (ウェブサイト内の他のページと同じ書式設定) のメッセージではなく、カスタムエラーメッセージを表示させたい場合は、そのカスタムエラーメッセージを含むオブジェクト (HTML ファイルなど) を、CloudFront からビューワーに返すようにもできます。

返信させたい特定のファイルや、ファイルの返信の対象となるエラーを指定するには、CloudFront ディストリビューションを更新してそれらの値を指定します。詳細については、「」を参照してくださいエラーレスポンスの動作を設定する

例として、カスタムエラーページは次のようなものになります。



サポートされている HTTP ステータスコードごとに異なるオブジェクトを指定することができます。または、サポートされているすべてのステータスコードに同じオブジェクトを使用することもできます。一部のステータスコードにカスタムエラーページを指定し、他のステータスコードには指定しないようにもできます。

CloudFront を通して供給されているオブジェクトは、様々な理由で使用できなくなることがあります。理由は、大きく 2 つに分類できます。以下にそれぞれを説明します。

- クライアントエラーは、リクエストに問題があることを示します。たとえば、指定した名前のオブジェクトが使用不能である場合や、Amazon S3 バケット内のオブジェクトを取得するために必要な権限がユーザーにない場合などです。クライアントエラーが発生すると、オリジンは 400 番台の HTTP ステータスコードを CloudFront に返します。
- サーバーエラーの場合は、オリジンサーバーに問題があります。たとえば、HTTP サーバーが混雑していたり使用不能であったりする場合です。サーバーエラーが発生すると、オリジンサーバーから 500 番台の HTTP ステータスコードが CloudFront に返されます。あるいは、オリジンサーバーから一定の時間内で CloudFront にレスポンスが届かないということで、504 ステータスコード (ゲートウェイタイムアウト) として処理されます。

CloudFront がカスタムエラーページを返すことのできる HTTP ステータスコードは、以下のとおりです。

- 400、403、404、405、414、416

> **メモ**
>
> - CloudFront は、リクエストが安全でない可能性があることを検出した場合、カスタムエラーページの代わりに 400 (不正なリクエスト) エラーを返します。
> - リクエストされた範囲は不適格であることを示す HTTP ステータスコード 416 (Requested Range Not Satisfiable) のカスタムエラーページを作成したり、オリジンが CloudFront にステータスコード 416 を返すと CloudFront がビューワーに返す HTTP ステータスコードを変更したりできます。(詳しくは、CloudFront から返されるレスポンスコードを変更する を参照してください)。ただし、CloudFront はステータスコード 416 のレスポンスをキャッシュしないため、ステータスコード 416 に対して [Error Caching Minimum TTL] (エラーキャッシュの最小 TTL) の値を指定しても、CloudFront はそれを使用しません。

- 500、501、502、503、504

> **Note**
>
> また、CloudFront で HTTP 503 ステータスコードのカスタムエラーページを設定していても、それが返されないケースもあり得ます。CloudFront エラーコードが `Capacity Exceeded` または `Limit Exceeded` の場合、CloudFront はカスタムエラーページを使用せずに 503 ステータスコードをビューワーに返します。

CloudFront がオリジンからのエラーレスポンスを処理する方法の詳細な説明は、「CloudFront がオリジンからの HTTP 4xx および 5xx ステータスコードを処理する方法」を参照してください。

オブジェクトとカスタムエラーページを別々の場所に保存する

オブジェクトとカスタムエラーページを別の場所に保存する場合は、次の状況に該当するときに適用されるキャッシュ動作をディストリビューションに組み込む必要があります。

- [Path Pattern (パスパターン)] の値が、カスタムエラーメッセージのパスと一致している。たとえば、4xx エラーのカスタムエラーページを `/4xx-errors` というディレクトリの Amazon S3 バケットに保存したとします。このとき、パスパターンによってカスタムエラーページのリクエストがルーティングされる場所のキャッシュ動作を、ディストリビューションに組み込む必要があります (`/4xx-errors/*` など)。

- [Origin (オリジン)] の値は、カスタムエラーページが含まれているオリジンの [Origin ID (オリジン ID)] の値を指定しています。

詳細については、「キャッシュ動作の設定」を参照してください。

CloudFront から返されるレスポンスコードを変更する

CloudFront がオリジンから受信したものとは異なる HTTP ステータスコードを、ビューワーに返すように CloudFront を設定できます。たとえば、オリジンから 500 ステータスコードが CloudFront に返されるときに、CloudFront からカスタムエラーページと 200 ステータスコード (OK) がビューワーに返されるようにしたいことがあります。さまざまな理由で、オリジンから CloudFront に返されるステータスコードとは異なるステータスコードが CloudFront からビューワーに返されることが必要になる場合があります。

- インターネットデバイス (一部のファイアウォールやコーポレートプロキシなど) の中には、HTTP 400 番台と 500 番台のステータスコードを遮断して、このレスポンスがビューワーに返信されないようするものがあります。このシナリオの場合、200 に置換することで、応答は遮断されなくなります。
- クライアントエラーとサーバーエラーの種類による区別が必要ない場合、CloudFront が返す 4xx および 5xx のステータスコードのすべてで、値を 400 または 500 とするようにも指定できます。
- 200 ステータスコード (OK) と静的ウェブサイトを返すことにより、ウェブサイトが停止していることをユーザーが気づかないようにもできます。

CloudFront 標準ログを有効にし、レスポンスの HTTP ステータスコードを変更するように CloudFront を設定すると、ログの sc-status 列の値には指定したステータスコードが記述されます。ただし、x-edge-result-type 列の値は影響を受けません。この列には、オリジンから返された結果タイプが記述されます。例えば、オリジンが 200 (見つかりません) を CloudFront に返す場合、404 のステータスコードをビューワーに返すように CloudFront を設定するとします。オリジンが 404 ステータスコードでリクエストに応答すると、ログ内の sc-status 列の値は 200 になりますが、x-edge-result-type 列の値は Error になります。

カスタムエラーページと共に以下の HTTP ステータスコードのいずれかを返すように、CloudFront を設定できます。

- 200
- 400、403、404、405、414、416
- 500、501、502、503、504

CloudFront がエラーをキャッシュする時間を制御する

CloudFront は、エラーレスポンスをデフォルト時間の 10 秒だけキャッシュします。その後、CloudFront はオブジェクトへの次のリクエストをオリジンに送信し、エラーの原因となった問題が解決されているかどうかと、リクエストしたオブジェクトが利用可能であるかどうかを確認します。

CloudFront がキャッシュする 4xx および 5xx ステータスコードそれぞれに対して、エラーキャッシュ期間 (エラーキャッシュ最小 TTL) を指定することができます。(詳細については、CloudFront がキャッシュする HTTP 4xx および 5xx ステータスコード を参照してください)。期間を指定する場合は、以下の点に注意してください。

- 短いエラーキャッシュ期間を指定すると、長い期間を指定した場合に比べて CloudFront からオリジンに転送されるリクエストの数が多くなります。この構成で 5xx エラーが発生すると、エラーの返信処理のために、オリジンで障害を起こした問題が悪化する可能性があります。
- オリジンがオブジェクトに関するエラーを返すと、CloudFront は、エラーキャッシュ期間が終了するまで、オブジェクトのリクエストにエラーレスポンスで応答するか、またはカスタムエラーページで応答します。長いエラーキャッシュ期間を指定するなら、オブジェクトが再び利用可能になった後も、CloudFront は長い間、リクエストにエラーレスポンスまたはカスタムエラーページで引き続き応答するかもしれません。

> **Note**
>
> リクエストされた範囲は不適格であることを示す HTTP ステータスコード 416 (Requested Range Not Satisfiable) のカスタムエラーページを作成したり、オリジンが CloudFront にステータスコード 416 を返すと CloudFront がビューワーに返す HTTP ステータスコードを変更したりできます。(詳しくは、CloudFront から返されるレスポンスコードを変更する を参照してください)。ただし、CloudFront はステータスコード 416 のレスポンスをキャッシュしないため、ステータスコード 416 に対して [Error Caching Minimum TTL] (エラーキャッシュの最小 TTL) の値を指定しても、CloudFront はそれを使用しません。

CloudFront でエラーをキャッシュする時間をオブジェクトごとに制御したい場合は、そのオブジェクトのエラーレスポンスに適切なヘッダーを追加するようにオリジンサーバーを設定できます。

オリジンが `Cache-Control: max-age` ディレクティブ、`Cache-Control: s-maxage` ディレクティブ、または `Expires` ヘッダーを追加した場合、CloudFront は、ヘッダーの値と [Error Caching

Minimum TTL] (エラーキャッシュの最小 TTL) の値を比較してより大きい値の時間だけ、エラーレスポンスをキャッシュします。

> **Note**
>
> `Cache-Control: max-age` および `Cache-Control: s-maxage` の値は、エラーページをフェッチするキャッシュ動作に設定されている [Maximum TTL] (最大 TTL) の値以下にする必要があります。

オリジンが他の `Cache-Control` ディレクティブまたはヘッダーを追加した場合、CloudFront は [Error Caching Minimum TTL] (エラーキャッシュの最小 TTL) の値の時間だけ、エラーレスポンスをキャッシュします。

オブジェクトに対する 4xx または 5xx ステータスコードの有効期限が、設定した待機時間よりも長く、さらにオブジェクトが使用可能状態に復帰した場合は、リクエストされたオブジェクトの URL を使ってそのステータスコードを無効化できます。オリジンが複数のオブジェクトに対してエラーレスポンスを返している場合は、各オブジェクトについて個別に無効化する必要があります。オブジェクトの無効化については、「ファイルを無効化してコンテンツを削除する」を参照してください。

CloudFront が配信するコンテンツを追加、削除、または置き換える

このセクションでは、CloudFront がビューワーに配信するコンテンツにアクセスできることを確認する方法、ウェブサイトやアプリケーションでオブジェクトを指定する方法、およびコンテンツを削除または置換する方法を説明します。

トピック

CloudFront が配信するコンテンツの追加とアクセス

CloudFront でコンテンツ (オブジェクト) を配信するときは、ディストリビューション用に指定したオリジンの 1 つにファイルを追加し、そのファイルに対する CloudFront リンクを公開します。CloudFront エッジロケーションは、ファイルに関するビューワーリクエストを受け取るまで、オリジンから新しいファイルを取得しません。詳細については、「CloudFront がコンテンツを配信する方法」を参照してください。

CloudFront で配信するファイルを追加するときは、ディストリビューションで指定した Amazon S3 バケットの 1 つに追加しているか、カスタムオリジンの場合は指定したドメイン内のディレクトリに追加していることを確認してください。さらに、該当するキャッシュ動作のパスパターンが正しいオリジンにリクエストを送信していることを確認します。

たとえば、キャッシュ動作のパスパターンが `*.html` であるとします。他のキャッシュ動作がない場合、CloudFront は、リクエストをそのオリジンに転送するように設定されている `*.html` ファイルのみを転送します。このシナリオでは、例えば .jpg ファイルが含まれているキャッシュ動作を作成していないため、そのオリジンにアップロードされた .jpg ファイルを CloudFront が配信することはありません。

CloudFront サーバーは、供給するオブジェクトの MIME タイプを判別しません。ファイルをオリジンにアップロードする場合、ファイルの Content-Type ヘッダーフィールドを設定することをお勧めします。

ファイルバージョニングを使用して CloudFront ディストリビューションでコンテンツを更新または削除する

CloudFront がユーザーに代わって配信するように設定されている既存のコンテンツを更新するには、ファイル名またはフォルダ名にバージョン識別子を使用することをお勧めします。これにより、CloudFront が提供するコンテンツの管理を制御できます。

バージョン付きのファイル名を使用して既存ファイルを更新する

CloudFront ディストリビューション内の既存のファイルを更新する場合、何らかのバージョン識別名をファイル名またはディレクトリ名に含めて、コンテンツを容易に制御できるようにすることをお勧めします。この識別名には、日付タイムスタンプ、連番など、同じオブジェクトの 2 つのバージョンを区別する方法を使用できます。

たとえば、グラフィックファイルに image.jpg ではなく image_1.jpg という名前を付けることができます。ファイルの新しいバージョンを供給する場合は、新しいファイルに image_2.jpg という名前を付けて、image_2.jpg を指すようにお使いのウェブアプリケーションまたはウェブサイトのリンクを更新します。また、すべてのグラフィックを images_v1 ディレクトリに配置することもできます。1 つ以上のグラフィックの新しいバージョンを供給する場合は、新しい images_v2 ディレクトリを作成し、そのディレクトリを指すようにリンクを更新します。バージョニングを行うことで、CloudFront で新しいバージョンの供給を開始するのにオブジェクトの有効期限切れを待つ必要がなくなり、オブジェクトの無効化に対して料金を支払う必要もなくなります。

ファイルにバージョンを設定した場合も、有効期限切れ日付を設定することをお勧めします。詳細については、「コンテンツをキャッシュに保持する期間 (有効期限) を管理する」を参照してください。

> **Note**
>
> バージョン付きのファイル名またはディレクトリ名の指定は、Amazon S3 オブジェクトのバージョニングとは関係がありません。

コンテンツを削除して CloudFront が配信しないようにする

CloudFront ディストリビューションに含めたくないファイルをオリジンのファイルから削除することができます。ただし、CloudFront はファイルの有効期限が切れるまで引き続きエッジキャッシュからコンテンツをビューワーに表示します。

すぐにファイルを削除する場合は、次のいずれかを実行する必要があります。

- ファイルバージョニングを使用します。バージョニングを使用する場合、異なるバージョンのファイルには、どのファイルがビューワーに返されるかを変更する CloudFront ディストリビューションで使用できる異なる名前があります。詳細については、「バージョン付きのファイル名を使用して既存ファイルを更新する」を参照してください。
- ファイルを無効化します。詳細については、「ファイルを無効化してコンテンツを削除する」を参照してください。

CloudFront でファイルの URL 形式をカスタマイズする

CloudFront がビューワーに提供するオブジェクト (コンテンツ) をオリジンに設定した後は、CloudFront がそれを提供できるように、ウェブサイトまたはアプリケーションコード内のオブジェクトを参照する正しい URL を使用する必要があります。

ウェブページやウェブアプリケーションのオブジェクトの URL で使用するドメイン名には、次のいずれかを指定できます。

- ディストリビューションを作成するときに CloudFront が自動的に割り当てる `d111111abcdef8.cloudfront.net` などのドメイン名
- `example.com` など、独自のドメイン名

たとえば、`image.jpg` というファイルを返すために、次の URL のいずれかを使用します。

```
https://d111111abcdef8.cloudfront.net/images/image.jpg
```

```
https://example.com/images/image.jpg
```

Amazon S3 バケット、またはカスタムオリジンのどちらにコンテンツを保存しても、独自のウェブサーバーと同様、同じ URL 形式を使用します。

> **Note**
>
> URL 形式は、ディストリビューションの [Origin Path] に指定した値によってある程度異なります。この値が CloudFront にオブジェクトの最上位ディレクトリパスを与えます。ウェブディストリビューションを作成する際のオリジンパスの設定の詳細については、「オリジンのパス」を参照してください。

URL 形式の詳細については、次のセクションを参照してください。

独自のドメイン名 (example.com) を使用する

ディストリビューションを作成するときに CloudFront が割り当てるデフォルトのドメイン名を使用する代わりに、example.com などの、より簡単に使用できる代替ドメイン名を追加できます。CloudFront で独自のドメイン名を設定すると、ディストリビューション内のオブジェクトにこのような URL を使用できます。

```
https://example.com/images/image.jpg
```

ビューワーと CloudFront の間で HTTPS を使用する予定がある場合は、「代替ドメイン名と HTTPS を使用する」を参照してください。

URL で末尾のスラッシュ (/) を使用する

CloudFront ディストリビューション内のディレクトリの URL を指定するとき、末尾のスラッシュを常に使用するか、または、全く使用しないかを選択します。たとえば、すべての URL に対して次のいずれか 1 つの形式のみを選択します。

```
https://d111111abcdef8.cloudfront.net/images/
```

```
https://d111111abcdef8.cloudfront.net/images
```

それが重要なのはなぜか。

どちらの形式でも CloudFront オブジェクトへのリンクとして機能しますが、一貫性を持たせることで、後でディレクトリを無効にするときに問題を防ぐことができます。CloudFront は、末尾のスラッシュも含め、指定されたとおりに URL を保存します。形式に一貫性がない場合、スラッシュのあるディレクトリ URL とスラッシュのないものを無効にして、CloudFront がディレクトリを削除したことを確認する必要があります。

両方の URL 形式を無効にするのは不便で、追加コストが発生することがあります。これは、両方のタイプの URL をカバーするために無効化を二重に実行する場合、1 か月の無料の無効化回数の上限に達する可能性があるためです。そして、もしそのような事態になれば、CloudFront の各ディレクトリ URL にはただ 1 つの形式しか存在しない場合でも、すべての無効化に対して支払いをする必要があります。

制限されたコンテンツの署名付き URL を作成する

アクセスを制限するコンテンツがある場合は、署名付き URL を作成できます。たとえば、認証されたユーザーのみにコンテンツを配信する場合は、指定された期間のみ、または指定された IP アドレスからのみ有効な URL を作成できます。詳細については、「署名付き URL と署名付き Cookie を使用したプライベートコンテンツを提供する」を参照してください。

デフォルトのルートオブジェクトを指定する

ユーザーがディストリビューション内のオブジェクトではなくディストリビューションのルート URL をリクエストした場合に特定のオブジェクト (デフォルトのルートオブジェクト) を返すように CloudFront を設定できます。デフォルトのルートオブジェクトを指定すると、ディストリビューションのコンテンツが公開されなくなります。

トピック

- デフォルトのルートオブジェクトを指定する方法
- デフォルトのルートオブジェクトの仕組み
- ルートオブジェクトを定義しない場合の CloudFront の動作

デフォルトのルートオブジェクトを指定する方法

ディストリビューションのコンテンツが公開されたり、エラーが返されたりすることを回避するには、以下のステップを実行して、ディストリビューションのデフォルトのルートオブジェクトを指定します。

ディストリビューションのデフォルトルートオブジェクトを指定するには

1. デフォルトルートオブジェクトを、ディストリビューションが指しているオリジンにアップロードします。

ファイルには、CloudFront でサポートされるあらゆるタイプを使用できます。ファイル名に対する制約事項のリストについては、「DefaultRootObjectDistributionConfig 」の エレメントの説明を参照してください。

> **Note**
>
> デフォルトルートオブジェクトのファイル名が長すぎるか、そのファイル名に無効な文字が含まれている場合、CloudFront はエラー HTTP 400 Bad Request - InvalidDefaultRootObject を返します。また、CloudFront はコードをキャッシュに (デフォルトで) 10 秒間保持し、結果をアクセスログに書き込みます。

2. オブジェクトのアクセス許可によって CloudFront に少なくとも read アクセスが付与されていることを確認します。

   Amazon S3 のアクセス権限の詳細については、Amazon Simple Storage Service ユーザーガイドの「Amazon S3 での Identity and Access Management」を参照してください。

3. CloudFront コンソールまたは CloudFront API を使用して、デフォルトルートオブジェクトを参照するようにディストリビューションを更新します。

   CloudFront コンソールを使用して、デフォルトルートオブジェクトを指定するには、次を実行します。

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

   b. 上部ペインにあるディストリビューションのリストで、更新するディストリビューションを選択します。

   c. [Settings] (設定) ペインで、[General] (一般) タブの [Edit] (編集) を選択します。

   d. [Edit settings] (設定の編集) ダイアログボックスの [Default root object] (デフォルトルートオブジェクト) フィールドに、デフォルトルートオブジェクトのファイル名を入力します。

      オブジェクト名のみを入力します (例: index.html)。オブジェクト名の前に / を追加しないでください。

   e. [Save changes] (変更の保存) をクリックします。

   CloudFront API を使用して、設定を更新するには、ディストリビューションの DefaultRootObject エレメントの値を指定します。CloudFront API を使用してデフォルト

のルートオブジェクトを指定する方法については、Amazon CloudFront API リファレンスの「UpdateDistribution」を参照してください。

4. ルート URL を要求することで、デフォルトルートオブジェクトが有効になっていることを確認します。ブラウザにデフォルトルートオブジェクトが表示されない場合、以下のステップを実行します。

   a. CloudFront コンソールにディストリビューションのステータスを表示し、ディストリビューションが完全にデプロイされていることを確認します。

   b. ステップ 2 と 3 を繰り返し、適切な許可を付与したこと、およびデフォルトルートオブジェクトを指定するようにディストリビューションの構成を適切に更新したことを確認します。

デフォルトのルートオブジェクトの仕組み

次のリクエストはオブジェクト `image.jpg` を指すとします。

```
https://d111111abcdef8.cloudfront.net/image.jpg
```

これに対して、最初の例のように、次のリクエストは特定のオブジェクトではなく、同じディストリビューションのルート URL を指します。

```
https://d111111abcdef8.cloudfront.net/
```

デフォルトルートオブジェクトを定義した場合、ディストリビューションのルートを呼び出すエンドユーザーリクエストはデフォルトルートオブジェクトを返します。たとえば、ファイル `index.html` をデフォルトルートオブジェクトとして指定したと仮定します。

```
https://d111111abcdef8.cloudfront.net/
```

戻り値:

```
https://d111111abcdef8.cloudfront.net/index.html
```

> **Note**
>
> CloudFront は、末尾に複数のスラッシュ (`https://d111111abcdef8.cloudfront.net///`) が付いている URL が `https://d111111abcdef8.cloudfront.net/` と等しいかどうかを判断しません。オリジンサーバーがその比較を行います。

デフォルトルートオブジェクトを定義しても、ディストリビューションのサブディレクトリに対するエンドユーザーリクエストはデフォルトルートオブジェクトを返しません。例えば、`index.html` がデフォルトルートオブジェクトであり、CloudFront が CloudFront ディストリビューション下の `install` ディレクトリに対するエンドユーザーリクエストを受け取ったと仮定します。

```
https://d111111abcdef8.cloudfront.net/install/
```

`index.html` のコピーが `install` ディレクトリ内にあっても、CloudFront はデフォルトルートオブジェクトを返しません。

CloudFront がサポートするすべての HTTP メソッドを許可するようにディストリビューションを設定する場合、デフォルトのルートオブジェクトがすべてのメソッドに適用されます。例えば、デフォルトのルートオブジェクトが index.php であり、POST リクエストをドメインのルート (https://example.com) に送信するようにアプリケーションを作成するなら、CloudFront はリクエストを https://example.com/index.php に送信します。

CloudFront のデフォルトルートオブジェクトの動作は、Amazon S3 のインデックスドキュメントの動作とは異なります。Amazon S3 バケットをウェブサイトとして設定し、インデックスドキュメントを指定した場合、ユーザーがバケット内のサブディレクトリを要求しても、Amazon S3 はインデックスドキュメントを返します。(インデックスドキュメントのコピーがすべてのサブディレクトリに含まれる必要があります)。Amazon S3 バケットをウェブサイトとして設定する方法とインデックスドキュメントの詳細については、Amazon Simple Storage Service ユーザーガイドの「Amazon S3 でのウェブサイトのホスティング」を参照してください。

> **Important**
>
> デフォルトルートオブジェクトは CloudFront ディストリビューションにのみ適用されることに注意してください。オリジンのセキュリティを依然として管理する必要があります。例えば、Amazon S3 オリジンを使用する場合、Amazon S3 バケットの ACL も依然として適切に設定する必要があり、バケットに対する必要なアクセスレベルを確保する必要があります。

ルートオブジェクトを定義しない場合の CloudFront の動作

デフォルトルートオブジェクトを定義しない場合、ディストリビューションのルートの要求はオリジンサーバーに渡されます。Amazon S3 オリジンを使用する場合、以下のいずれかが返される場合があります。

- Amazon S3 バケットのコンテンツのリスト – 以下のいずれかの条件が満たされる場合、CloudFront を使用してディストリビューションにアクセスするユーザーにオリジンのコンテンツが表示されます。
  - バケットが適切に構成されていない。
  - ディストリビューションに関連付けられているバケット、およびバケット内のオブジェクトに対する Amazon S3 アクセス許可で、すべてのユーザーにアクセスが付与されている。
  - エンドユーザーがオリジンのルート URL を使用してオリジンにアクセスしている。
- オリジンのプライベートコンテンツのリスト – オリジンをプライベートディストリビューション (設定者と CloudFront のみがアクセス可能) として設定している場合、ディストリビューションに関連付けられた Amazon S3 バケットのコンテンツは、CloudFront 経由でディストリビューションにアクセスできる認証情報を持っているユーザーにも表示されます。この場合、ユーザーは、オリジンのルート URL を使用してコンテンツにアクセスできません。プライベートコンテンツの配信の詳細については、「the section called “署名付き URL と署名付き Cookie を使用したコンテンツを制限する”」を参照してください。
- `Error 403 Forbidden` – CloudFront は、ディストリビューションに関連付けられた Amazon S3 バケットに対する許可またはそのバケット内のオブジェクトに対する許可によって CloudFront およびすべてのユーザーのアクセスが拒否された場合にこのエラーを返します。

ファイルを無効化してコンテンツを削除する

有効期限切れになる前に CloudFront エッジキャッシュからファイルを削除する場合、以下のいずれかの処理を行うことができます。

- エッジキャッシュからファイルを無効にします。ビューワーが次にファイルをリクエストしたときに、CloudFront はオリジンに戻ってファイルの最新バージョンをフェッチします。
- ファイルのバージョニングを使用して、異なる名前を持つ異なるバージョンのファイルを供給します。詳細については、「バージョン付きのファイル名を使用して既存ファイルを更新する」を参照してください。

トピック

- ファイルを無効化するか、バージョン付きファイル名を使用するかを選択する
- 無効にするファイルを決定する
- ファイルを無効にするときに知っておくべきこと
- ファイルを無効化する

- 同時無効化リクエストの最大制限
- ファイルの無効化に対する支払い

ファイルを無効化するか、バージョン付きファイル名を使用するかを選択する

ディストリビューションから供給されるファイルのバージョンを制御するには、ファイルを無効にするか、バージョン付きファイル名をファイルに設定します。ファイルを頻繁に更新する必要がある場合は、以下の理由で、ファイルのバージョニングを第一に使用することをお勧めします。

- バージョニングを使用すると、ローカルにキャッシュされている、または企業のキャッシュプロキシの背後にキャッシュされているバージョンをユーザーが保持している場合でも、リクエストがどのファイルを返すかを制御できます。ファイルを無効にした場合、キャッシュ内でオブジェクトが有効期限切れになるまで、ユーザーに旧バージョンが引き続き表示されることがあります。
- CloudFront アクセスログにファイル名が含まれるので、バージョニングを使用すると、ファイルの変更結果の分析が容易になります。
- バージョニングは、さまざまなバージョンのファイルをさまざまなユーザーに供給する方法を提供します。
- バージョニングによって、ファイルのリビジョン間のロールフォワードとロールバックが簡素化されます。
- バージョニングのほうが、コストが安くなります。CloudFront が新しいバージョンのファイルをエッジロケーションに転送することに関して料金を支払う必要がありますが、ファイルの無効化に関して料金を支払う必要はありません。

ファイルのバージョニングの詳細については、バージョン付きのファイル名を使用して既存ファイルを更新する を参照してください。

無効にするファイルを決定する

ディレクトリ内のすべてのファイルや、名前が同じ文字で始まるすべてのファイルなど、複数のファイルを無効にする場合は、無効化パスの末尾に * ワイルドカードを含めることができます。* ワイルドカードの使用の詳細については、「Invalidation paths」を参照してください。

ファイルを無効にするには、個々のファイルのパスまたは * ワイルドカードで終わるパスのいずれかを指定します。これは次の例に示すように、1 つまたは複数のファイルに適用できます。

- /images/image1.jpg
- /images/image*
- /images/*

選択されたファイルを無効にする必要があり、ユーザーがオリジンのすべてのファイルに必ずしもアクセスしない場合は、ビューワーが CloudFront からどのファイルを要求したかを確認し、そのファイルのみを無効にできます。ビューワーがどのファイルを要求したかを確認するには、CloudFront アクセスログの作成を有効にします。アクセスログの詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

ファイルを無効にするときに知っておくべきこと

無効にするファイルを指定するときには、次の情報を参照してください。

大文字と小文字の区別

無効化パスでは大文字と小文字が区別されます。例えば、/images/image.jpg と /images/Image.jpg は 2 つの異なるファイルを指定します。

Lambda 関数を使用した URI の変更

CloudFront ディストリビューションがビューワーリクエストイベントで Lambda 関数をトリガーして、関数がリクエストされたファイルの URI を変更する場合、CloudFront エッジキャッシュからファイルを削除するために両方の URI を無効にすることを推奨します。

- ビューワーリクエストの URI
- 関数による変更後の URI

Example 例

Lambda 関数が以下のファイルの URI を、

https://d111111abcdef8.cloudfront.net/index.html

言語ディレクトリを含む URI に変更するとします。

https://d111111abcdef8.cloudfront.net/en/index.html

ファイルを無効にするには、次のパスを指定する必要があります。

- /index.html
- /en/index.html

詳しくは、「Invalidation paths」を参照してください。

デフォルトのルートオブジェクト

デフォルトルートオブジェクト (ファイル) を無効にする場合、他のファイルのパスを指定する場合と同じ方法でパスを指定します。詳細については、「デフォルトのルートオブジェクトの仕組み」を参照してください。

Cookie の転送

オリジンに Cookie を転送するように CloudFront を設定した場合、CloudFront エッジキャッシュにファイルの複数のバージョンが含まれることがあります。ファイルを無効にすると、CloudFront は、関係付けられた Cookie に関係なく、そのファイルのキャッシュされたあらゆるバージョンを無効にします。一部のバージョンを選択して無効にすることも、関連付けられた Cookie に基づいてその他のバージョンを選択して無効にすることもできません。詳しくは、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。

ヘッダーの転送

ヘッダーのホワイトリストをオリジンに転送し、ヘッダーの値に基づいてキャッシュするように CloudFront を設定した場合、CloudFront エッジキャッシュにファイルの複数のバージョンが含まれることがあります。ファイルを無効にすると、CloudFront は、ヘッダー値に関係なく、そのファイルのキャッシュされたあらゆるバージョンを無効にします。ヘッダー値に基づいて一部のバージョンのみ選択して無効にすることはできません (すべてのヘッダーをオリジンに転送するように CloudFront を設定した場合、CloudFront はファイルをキャッシュしません)。詳しくは、「リクエストヘッダーに基づいてコンテンツをキャッシュする」を参照してください。

クエリ文字列の転送

クエリ文字列をオリジンに転送するように CloudFront を設定している場合、次の例に示すように、ファイルを無効にするときにクエリ文字列を含める必要があります。

- `/images/image.jpg?parameter1=a`
- `/images/image.jpg?parameter1=b`

クライアントリクエストに、同じファイルに対する 5 つの異なるクエリ文字列が含まれる場合、クエリ文字列ごとに 1 回ずつ、5 回を無効にするか、次の例に示すように個別の無効化パスに * ワイルドカードを使用できます。

```
/images/image.jpg*
```

無効化パスでのワイルドカードの使用の詳細については、「Invalidation paths」を参照してください。

クエリ文字列の詳細については、「クエリ文字列パラメータに基づいてコンテンツをキャッシュする」を参照してください。

使用中のクエリ文字列を確認するには、CloudFront ログ作成を有効にすることができます。詳しくは、「標準ログ (アクセスログ) の設定および使用」を参照してください。

最大許容数

許可される無効化の最大許容値については、「同時無効化リクエストの最大制限」を参照してください。

Microsoft Smooth Streaming ファイル

対応するキャッシュ動作に対してスムーズストリーミングを有効にした場合は、Microsoft Smooth Streaming 形式のメディアファイルを無効にすることはできません。

パス内の ASCII 以外の文字または安全ではない文字

パスに ASCII 以外の文字が含まれるか、RFC 1738 に規定された安全ではない文字が含まれる場合、その文字を URL エンコードします。パスに含まれる他の文字を URL エンコードしないでください。URL エンコードした場合、CloudFront は、更新されたファイルの旧バージョンを無効にしません。

無効化パス

パスはディストリビューションを基準とする相対パスです。例えば、`https://d111111abcdef8.cloudfront.net/images/image2.jpg` のファイルを無効にするには、`/images/image2.jpg` を指定します。

> **Note**
>
> CloudFront コンソールでは、`images/image2.jpg` のようにパスの先頭のスラッシュを省略できます。CloudFront API を直接使用する場合、無効化パスは先頭にスラッシュを付ける必要があります。

* ワイルドカードを使用して、同時に複数のファイルを無効にすることもできます。0 個以上の文字を置き換える * は、無効化パスの最後の文字である必要があります。

ファイルの無効化に AWS Command Line Interface (AWS CLI) を使用し、* ワイルドカードが含まれるパスを指定する場合は、"/*" のようにパスを引用符 (") で囲む必要があります。

Example 例: 無効化パス

- ディレクトリ内のすべてのファイルを無効にするには:

  /*directory-path*/*

- ディレクトリ、そのすべてのサブディレクトリ、およびそのディレクトリとサブディレクトリのすべてのファイルを無効にするには:

  /*directory-path**

- 同じ名前でファイル名拡張子が異なるすべてのファイル (logo.jpg、logo.png、logo.gif など) を無効にするには:

  /*directory-path*/*file-name*.*

- ファイル名拡張子にかかわらず、ディレクトリ内でファイル名が同じ文字で始まるすべてのファイル (HLS 形式の動画のすべてのファイルなど) を無効にするには:

  /*directory-path*/*initial-characters-in-file-name**

- クエリ文字列パラメータに基づいてキャッシュするように CloudFront を設定し、ファイルのすべてのバージョンを無効にするには:

  /*directory-path*/*file-name*.*file-name-extension**

- ディストリビューション内のすべてのファイルを無効にするには:

  /*

パスの最大長は 4000 文字です。パス内でワイルドカードを使用することはできません。パスの末尾にのみ追加できます。

Lambda 関数を使用して URI を変更する場合のファイルの無効化の詳細については、「Changing the URI Using a Lambda Function」を参照してください。

無効化パスがディレクトリであり、ディレクトリの指定方法 (末尾のスラッシュ (/) を付けるかどうか) を標準化していない場合、末尾のスラッシュを付けたディレクトリと付けないディレクトリの両方を無効にすることをお勧めします (例: `/images` および `/images/`)。

書名付き URL

署名付き URL を使用している場合は、URL の疑問符 (?) の前の部分のみを含めてファイルを無効にします。

ファイルを無効化する

CloudFront コンソールを使用して、無効化の作成と実行、以前に送信された無効化のリストの表示、および個々の無効化に関する詳細情報の表示を行うことができます。また、既存の無効化をコピーしたり、ファイルパスのリストを編集したり、編集された無効化を実行したりもできます。無効化をリストから削除することはできません。

目次

ファイルを無効化する

CloudFront コンソールを使用してファイルを無効にするには、以下を実行します。

Console

ファイルを無効化するには (コンソール)

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 無効にするファイルのディストリビューションを選択します。
3. [Invalidations] タブを選択します。
4. [Create Invalidation] を選択します。
5. 無効にするファイルについて、1 行ごとに 1 つの無効化パスを入力します。無効パスの指定の詳細については、「ファイルを無効にするときに知っておくべきこと」を参照してください。

> **Important**
>
> ファイルパスを慎重に指定します。無効化リクエストは開始後にキャンセルすることはできません。

6. [Create Invalidation] を選択します。

CloudFront API

オブジェクトの無効化と、無効化に関する情報の表示については、「Amazon CloudFront API リファレンス」の以下のトピックを参照してください。

- CreateInvalidation
- ListInvalidations
- GetInvalidation

> **Note**
>
> ファイルの無効化に AWS Command Line Interface (AWS CLI) を使用し、* ワイルドカードが含まれるパスを指定する場合は、以下の例のようにパスを引用符 (") で囲む必要があります。
>
> ```
> aws cloudfront create-invalidation --distribution-id distribution_ID --paths
>  "/*"
> ```

既存の無効化のコピー、編集、および再実行を行う

以前に作成した無効化をコピーし、無効化パスのリストを更新して、更新した無効化を実行することができます。既存の無効化をコピーし、無効化パスを更新して、更新した無効化を実行せずに保存することはできません。

> **Important**
>
> 進行中の無効化をコピーし、無効化パスのリストを更新して、更新した無効化を実行した場合、コピーした無効化を CloudFront が停止または削除することはありません。ある無効化パスがファイルとコピーの両方に含まれる場合、CloudFront はこのファイルの無効化を 2 回試みます。この 2 回の無効化は月ごとの無料の無効化の最大数に対してカウントされます。無料で行うことができる無効化の最大数に既に達している場合は、各ファイルの両方の無効化に対して料金が発生します。詳細については、「同時無効化リクエストの最大制限」を参照してください。

既存の無効化のコピー、編集、および再実行を行うには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. コピーする無効化が含まれるディストリビューションを選択します。
3. [Invalidations] タブを選択します。
4. コピーする無効化を選択します。

   コピーする無効化が不明な場合は、無効化を選択し、[View details] を選択すると、その無効化の詳細情報が表示されます。
5. [新規にコピー] を選択します。
6. 必要に応じて、無効化パスのリストを更新します。
7. [Create Invalidation] を選択します。

無効化をキャンセルする

CloudFront に無効化リクエストを送信すると、そのリクエストは数秒以内にすべてのエッジロケーションに転送されて、各エッジロケーションで直ちに無効化の処理が開始されます。そのため、無効化を送信後にキャンセルすることはできません。

無効化のリストを表示する

CloudFront コンソールを使用して、ディストリビューションにおいて作成および実行された最後の 100 個の無効化のリストを表示できます。100 個を超える無効化のリストを取得する場合は、`ListInvalidations` API 操作を使用します。詳細については、Amazon CloudFront API リファレンスの「ListInvalidations」を参照してください。

無効化のリストを表示するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 無効化リストの表示の対象となるディストリビューションを選択します。
3. [Invalidations] タブを選択します。

> **Note**
>
> 無効化をリストから削除することはできません。

無効化に関する情報を表示する

ディストリビューション ID、無効化 ID、無効化のステータス、無効化が作成された日時、無効化パスの完全リストを含め、無効化に関する詳細情報を表示できます。

無効化に関する情報を表示するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 詳細情報の表示の対象となる無効化が含まれるディストリビューションを選択します。
3. [Invalidations] タブを選択します。
4. 該当する無効化 ID を選択するか、無効化 ID を選択してから、[詳細を表示] を選択します。

同時無効化リクエストの最大制限

ファイルを個別に無効にする場合は、進行中のディストリビューションごとに最大 3,000 個のファイルまで、一度に無効化リクエストを作成できます。これは、最大 3,000 個のファイルに対する 1 つの無効化リクエスト、1 つのファイルに対する最大 3,000 個のリクエスト、または 3,000 個のファイルを超えないその他の任意の組み合わせとすることができます。たとえば、それぞれ 100 個のファイルを無効にする 30 個の無効化リクエストを送信できます。30 個の無効化リクエストがすべてまだ実行中である限り、それ以上の無効化リクエストを送信することはできません。この最大制限を超えた場合、CloudFront はエラーメッセージを返します。

* ワイルドカードを使用している場合、最大 15 個の無効化パスのリクエストを一度に作成できます。また、進行中のディストリビューションごとに最大 3,000 個の個別のファイルを同時に作成することができます。ワイルドカードの無効化リクエストの最大制限は、ファイルの個別の無効化の最大制限とは無関係です。

ファイルの無効化に対する支払い

1 か月に送信した無効化パスのうち、最初の 1,000 件は無料です。1 か月に 1,000 件を超えると、無効化パス 1 件ごとに支払いが発生します。無効なパスは 1 つのファイル (`/images/logo.jpg` な

ど) に対して、または複数のファイル (/images/*) に対して発生する場合があります。CloudFront が無数のファイルを無効にした場合でも、ワイルドカード * は 1 つのパスとして含まれます。

1 か月あたり 1,000 個の無料の無効化パスの上限は、1 つの AWS アカウントで作成するすべてのディストリビューションの無効化パスの合計数に対して適用されます。例えば、AWS アカウント john@example.com を使用して 3 個のディストリビューションを作成し、ある月に、各ディストリビューションに 600 個の無効化パス (合計で 1,800 個の無効化パス) を送信した場合、AWS は、その月に 800 個の無効化パスに対して料金を請求します。

無効化パスを送信する料金は、無効にするファイルの数に関係なく同じです。つまり、1 つのファイル (/images/logo.jpg) であっても、ディストリビューションに関連付けられたすべてのファイル (/*) であっても同じです。無効化リクエストではパスごとに課金されるため、複数のパスを 1 つのリクエストにまとめた場合でも、請求の目的においては各パスが依然として個別にカウントされます。

無効化の料金に関する詳細情報については、「Amazon CloudFront 料金表」を参照してください。無効化パスの詳細については、「Invalidation paths」を参照してください。

圧縮ファイルを供給する

CloudFront を使用して、特定のタイプのオブジェクト (ファイル) を自動的に圧縮し、ビューワー (ウェブブラウザやその他のクライアント) でサポートされている場合は、その圧縮オブジェクトを供給できます。ビューワーが Accept-Encoding HTTP ヘッダーを含む圧縮オブジェクトのサポートの可否を示します。

CloudFront では、Gzip および Brotli 圧縮形式を使用してオブジェクトを圧縮できます。ビューワーが両方の形式をサポートしていて、アクセス先のキャッシュサーバーに両方が存在する場合、CloudFront は Brotli を優先します。キャッシュサーバーに圧縮形式が 1 つしかない場合、CloudFront はそれを返します。

> **Note**
>
> ウェブブラウザ Chrome および Firefox では、HTTPS を使用してリクエストを送信する場合のみ、Brotli 圧縮がサポートされます。これらのブラウザでは、HTTP リクエストで Brotli がサポートされません。

リクエストされたオブジェクトを圧縮するとオブジェクトが小さくなるため、ダウンロード時間を短縮できます。場合によっては、元のサイズの 4 分の 1 未満になることがあります。特に、JavaScript

および CSS ファイルでは、ダウンロードが速くなると、ユーザーにウェブページが表示されるまでの時間が短縮されます。また、CloudFront のデータ転送コストは供給されたデータの総量に基づくため、圧縮オブジェクトを処理する方が、非圧縮オブジェクトを供給するよりもコストが安くなる可能性があります。

一部のカスタムオリジンでは、オブジェクトを圧縮することもできます。オリジンでは、CloudFront で圧縮されないオブジェクトを圧縮できる場合があります (「CloudFront が圧縮するファイルタイプ」を参照)。オリジンが圧縮オブジェクトを CloudFront に返す場合、CloudFront ではオブジェクトが `Content-Encoding` ヘッダーの存在に基づいて圧縮されていることを検出し、オブジェクトを再度圧縮することはありません。

オブジェクトを圧縮するように CloudFront を設定する

オブジェクトを圧縮するように CloudFront を設定するには、次のすべての操作を実行して、圧縮オブジェクトを供給するキャッシュ動作を更新します。

1. [オブジェクトを自動的に圧縮する] 設定がはいになっていることを確認します。(AWS CloudFormation または CloudFront API で、`Compress` を `true` に設定します)。
2. キャッシュポリシーを使用してキャッシュ設定を指定し、Gzip と Brotli の設定がどちらも有効になっていることを確認します。(AWS CloudFormation または CloudFront API で、`EnableAcceptEncodingGzip` と `EnableAcceptEncodingBrotli` を `true` に設定します)。
3. キャッシュポリシーの TTL 値が 0 より大きい値に設定されていることを確認します。TTL 値をゼロに設定すると、キャッシュは無効になり、CloudFront によるオブジェクトの圧縮は行われません。

キャッシュ動作を更新するには、次のいずれかのツールを使用できます。

- CloudFront コンソール
- AWS CloudFormation
- AWS SDK とコマンドラインツール

CloudFront 圧縮の仕組み

オブジェクトが圧縮されるように CloudFront を設定した場合 (前のセクションを参照)の圧縮の仕組みをご紹介します。

1. ビューワーがオブジェクトを要求する ビューワーにより Accept-Encoding HTTP ヘッダーがリクエストに含められます。ヘッダー値には gzip、br、またはその両方が含められます。これは、ビューワーが圧縮オブジェクトをサポートすることを示します。ビューワーが Gzip と Brotli の両方をサポートしている場合、CloudFront では Brotli が優先されます。

   > **Note**
   >
   > ウェブブラウザ Chrome および Firefox では、HTTPS を使用してリクエストを送信する場合のみ、Brotli 圧縮がサポートされます。これらのブラウザでは、HTTP リクエストで Brotli がサポートされません。

2. CloudFront では、エッジロケーションで、リクエストされたオブジェクトの圧縮コピーのキャッシュを確認します。
3. 圧縮オブジェクトがすでにキャッシュにある場合、CloudFront ではそのオブジェクトをビューワーに送信し、残りの手順をスキップします。

   圧縮オブジェクトがキャッシュにない場合、CloudFront がリクエストをオリジンに転送します。

   > **Note**
   >
   > オブジェクトの非圧縮のコピーがすでにキャッシュにある場合、CloudFront はリクエストをオリジンに転送せずにビューワーに送信することがあります。例えば、CloudFront が以前に圧縮をスキップした場合に、この現象が起こる可能性があります。この場合、CloudFront では非圧縮オブジェクトをキャッシュし、オブジェクトが有効期限切れになるか、削除されるか、あるいは無効になるまで、処理を継続します。

4. オリジンが圧縮オブジェクトを返す場合 (HTTP レスポンスに Content-Encoding ヘッダーが存在することでわかります)、CloudFront は圧縮オブジェクトをビューワーに送信し、それをキャッシュに追加して、残りのステップをスキップします。CloudFront では、オブジェクトを再度圧縮することはありません。

   オリジンが CloudFront に非圧縮オブジェクトを返す場合 (HTTP レスポンスに Content-Encoding ヘッダーがない) 、CloudFront ではオブジェクトが圧縮可能かどうかを判断します。CloudFront でオブジェクトが圧縮可能かどうかを判断する方法の詳細については、次のセクションをご覧ください。
5. オブジェクトが圧縮可能な場合、CloudFront ではオブジェクトを圧縮して、それをビューワーに送信し、キャッシュに追加します (まれに、CloudFront は圧縮をスキップし、圧縮されていないオブジェクトをビューアに送信することがあります)。

CloudFront がオブジェクトを圧縮する場合

CloudFront でオブジェクトを圧縮するタイミングの詳細については、次のリストを参照してください。

リクエストで HTTP 1.0 が使用される

CloudFront へのリクエストで HTTP 1.0 が使用される場合、CloudFront では `Accept-Encoding` ヘッダーを削除し、レスポンスのオブジェクトを圧縮しません。

`Accept-Encoding` リクエストヘッダー

ビューワーリクエストに `Accept-Encoding` ヘッダーがない場合、またはヘッダーに `gzip` または `br` が値として含まれていない場合、CloudFront ではレスポンスのオブジェクトを圧縮しません。`Accept-Encoding` ヘッダーに追加の値 (`deflate` など) が含まれる場合、CloudFront ではそれらを削除してからリクエストをオリジンに転送します。

CloudFront がオブジェクトを圧縮するように設定されている場合、キャッシュキーとオリジンリクエストに `Accept-Encoding` ヘッダーが自動的に含められます。

動的コンテンツ

CloudFront では、必ずしも動的コンテンツを圧縮するとは限りません。動的コンテンツのレスポンスが圧縮される場合もあれば、圧縮されない場合もあります。

オブジェクトが圧縮されるように CloudFront を設定すると、コンテンツがすでにキャッシュされている

CloudFront では、オブジェクトをオリジンから取得したときにそのオブジェクトを圧縮します。オブジェクトが圧縮されるように CloudFront を設定した場合、CloudFront では、すでにエッジロケーションでキャッシュされたオブジェクトを圧縮しません。さらに、キャッシュされたオブジェクトがエッジロケーションで有効期限切れになり、CloudFront でオブジェクトに対する別のリクエストをオリジンに転送した場合に、HTTP ステータスコード 304 (エッジロケーションにすでにオブジェクトの最新バージョンがあることを意味する) がオリジンから返されると、CloudFront ではファイルを圧縮しません。エッジロケーションですでにキャッシュされているオブジェクトを CloudFront で圧縮するには、これらのオブジェクトを無効にする必要があります。詳細については、「ファイルを無効化してコンテンツを削除する」を参照してください。

オブジェクトが圧縮されるようにオリジンがすでに設定されている

オブジェクトが圧縮されるように CloudFront を設定し、オリジンでもオブジェクトが圧縮される場合、オリジンにはオブジェクトがすでに圧縮されていることを CloudFront に示す `Content-`

Encoding ヘッダーが含まれている必要があります。オリジンからのレスポンスに Content-Encoding ヘッダーが含まれる場合、ヘッダーの値に関係なく CloudFront ではオブジェクトを圧縮しません。CloudFront ではレスポンスをビューワーに送信し、エッジロケーションでオブジェクトをキャッシュします。

CloudFront が圧縮するファイルタイプ

CloudFront で圧縮するファイルタイプの詳細な一覧については、「CloudFront が圧縮するファイルタイプ」を参照してください。

CloudFront で圧縮するオブジェクトのサイズ

CloudFront では サイズが 1,000 ~ 10,000,000 バイトのオブジェクトを圧縮します。

Content-Length ヘッダー

オリジンは、レスポンスに Content-Length ヘッダーを含める必要があります。これは、オブジェクトのサイズが CloudFront で圧縮できる範囲にあるかどうかを判断するために使用されるものです。Content-Length ヘッダーがない、無効な値が含まれている、または CloudFront で圧縮できるサイズの範囲外の値が含まれている場合、CloudFront はオブジェクトを圧縮しません。

レスポンスの HTTP ステータスコード

CloudFront では、レスポンスの HTTP ステータスコードが 200、403、または 404 の場合にのみ、オブジェクトが圧縮されます。

レスポンスに本文がない

オリジンからの HTTP レスポンスに本文がない場合、CloudFront で圧縮するものはありません。

ETag ヘッダー

CloudFront では、オブジェクトを圧縮するときに HTTP レスポンスの ETag ヘッダーを変更することがあります。詳細については、「the section called “ETag ヘッダーの変換”」を参照してください。

CloudFront が圧縮をスキップする

CloudFront はベストエフォートベースでオブジェクトを圧縮します。まれに、CloudFront は圧縮をスキップすることがあります。CloudFront は、ホスト容量を含むさまざまな要因に基づいて、この決定を行います。CloudFront がオブジェクトの圧縮をスキップした場合、非圧縮オブジェクトをキャッシュし、オブジェクトが有効期限切れになるか、削除されるか、無効になるまで、処理を継続します。

CloudFront が圧縮するファイルタイプ

オブジェクトが圧縮されるように CloudFront を設定すると、CloudFront が `Content-Type` レスポンスヘッダーに次の値を持つオブジェクトだけを圧縮します。

- `application/dash+xml`
- `application/eot`
- `application/font`
- `application/font-sfnt`
- `application/javascript`
- `application/json`
- `application/opentype`
- `application/otf`
- `application/pdf`
- `application/pkcs7-mime`
- `application/protobuf`
- `application/rss+xml`
- `application/truetype`
- `application/ttf`
- `application/vnd.apple.mpegurl`
- `application/vnd.mapbox-vector-tile`
- `application/vnd.ms-fontobject`
- `application/wasm`
- `application/xhtml+xml`
- `application/xml`
- `application/x-font-opentype`
- `application/x-font-truetype`
- `application/x-font-ttf`
- `application/x-httpd-cgi`
- `application/x-javascript`
- `application/x-mpegurl`

- application/x-opentype
- application/x-otf
- application/x-perl
- application/x-ttf
- font/eot
- font/opentype
- font/otf
- font/ttf
- image/svg+xml
- text/css
- text/csv
- text/html
- text/javascript
- text/js
- text/plain
- text/richtext
- text/tab-separated-values
- text/xml
- text/x-component
- text/x-java-source
- text/x-script
- vnd.apple.mpegurl

ETag ヘッダーの変換

オリジンからの非圧縮オブジェクトに有効な強い ETag HTTP ヘッダーが含まれていて、CloudFront でそのオブジェクトを圧縮する場合、CloudFront では強い ETag ヘッダー値を弱い ETag に変換し、ビューワーに弱い ETag 値を返します。ビューワーは、弱い ETag 値を格納し、それを使用して If-None-Match HTTP ヘッダーで条件付きリクエストを送信できます。これにより、ビューワー、CloudFront、およびオリジンは、オブジェクトの圧縮バージョンと非圧縮バージョンを意味的に同等なものとして扱い、不要なデータ転送を減らすことができます。

有効な強い ETag ヘッダー値は、二重引用符 (") で始まります。強い ETag 値を弱い値に変換するために、CloudFront は強い W/ 値の先頭に文字 ETag を追加します。

オリジンからのオブジェクトに弱い ETag ヘッダー値 (文字 W/ で始まる値) が含まれている場合、CloudFront はこの値を変更せず、オリジンから受け取ったままビューワーに返します。

オリジンからのオブジェクトに無効な ETag ヘッダー値が含まれている場合 (値が " または W/ で始まらない)、CloudFront は ETag ヘッダーを削除し、ETag レスポンスヘッダーなしでオブジェクトをビューワーに返します。

詳細については、MDN ウェブドキュメントの以下のページを参照してください。

- ディレクティブ (ETag HTTP ヘッダー)
- 弱い検証 (HTTP 条件付きリクエスト)
- If-None-Match HTTP ヘッダー

AWS WAF 保護を使用する

CloudFront ディストリビューションとオリジンサーバーを保護するには、AWS WAF を使用できます。AWS WAF は、リクエストがサーバーに到達する前にブロックすることで、ウェブアプリケーションと API を保護するウェブアプリケーションファイアウォールです。詳細については、「CloudFront と AWS WAF を使用したウェブサイトの高速化と保護」を参照してください。

AWS WAF 保護を有効にするには、次のことができます。

- CloudFront コンソールでワンクリック保護を使用します。ワンクリック保護は、AWS WAF ウェブアクセスコントロールリスト (ウェブ ACL) を作成し、一般的なウェブの脅威からサーバーを保護するルールを設定して、ウェブ ACL を CloudFront ディストリビューションに自動的にアタッチします。このセクションのトピックでは、ワンクリック保護の使用を前提としています。
- AWS WAF コンソールまたは AWS WAF API を使用して作成した事前設定済みのウェブ ACL (アクセス制御リスト) を使用します。詳細については、「AWS WAF 開発者ガイド」の「ウェブアクセスコントロールリスト (ACL)」と「AWS WAF API リファレンス」の「AssociateWebACL」を参照してください。

以下の場合に AWS WAF を有効にすることができます。

- ディストリビューションを作成する
- [セキュリティ] ダッシュボードを使用して、既存のディストリビューションのセキュリティ設定を編集します。

ワンクリック保護を使用すると、CloudFront は次のような AWS が推奨する一連の保護を適用します。

- Amazon の内部脅威インテリジェンスに基づく潜在的な脅威から IP アドレスをブロックします。
- OWASP Top 10 で説明されているように、ウェブアプリケーションに見られる最も一般的な脆弱性から保護します。
- 悪意のある攻撃者がアプリケーションの脆弱性を発見するのを防ぎます。

> **Important**
>
> CloudFront の [セキュリティ] ダッシュボードにセキュリティメトリクスを表示するには、AWS WAF を有効にする必要があります。AWS WAF を有効にしない場合、[セキュリ

ティ] ダッシュボードでできることは AWS WAF を有効にするか、CloudFront の地理的制限を設定することだけです。ダッシュボードの詳細については、このセクションの後半にある「CloudFront セキュリティダッシュボードで AWS WAF セキュリティ保護を管理する」を参照してください。

トピック

ディストリビューションで AWS WAF を有効にする

ディストリビューションの作成時に AWS WAF を有効にすることも、既存のアクセスコントロールリスト (ACL) のセキュリティ保護を有効にすることもできます。

CloudFront ディストリビューションで AWS WAF を有効にすると、Bot Control を有効にしてボットカテゴリ別にセキュリティ保護を設定することもできます。

トピック

新しいディストリビューションで AWS WAF を有効にする

次の手順では、CloudFront ディストリビューションの作成時に AWS WAF を有効にする方法を示します。

新しいディストリビューションで AWS WAF を有効にするには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。
2. ナビゲーションペインで [ディストリビューション] を選択し、[ディストリビューションを作成] を選択します。

3. 必要に応じて、「ディストリビューションを作成する」の手順に従います。
4. [ウェブアプリケーションファイアウォール] セクションで、[編集]、[セキュリティ保護を有効にする] の順に選択します。
5. 以下のフィールドに値を入力します。

   - モニタリングモードを使う – 保護の仕組みをテストするために最初にデータを収集する場合は、モニタリングモードを有効にします。モニタリングモードを有効にすると、保護が有効になっていてもリクエストはブロックされません。代わりに、モニタリングモードは、保護が有効な場合にブロックされるリクエストに関するデータを収集します。ブロックを開始する準備ができたら、[セキュリティ] ページでブロックを有効にすることができます。
   - 追加保護 – 有効にするオプションを選択します。レート制限を有効にする場合の詳細については、「the section called “レート制限の設定”」を参照してください。
   - 価格見積もり – このセクションを開くと、1 か月あたりの異なるリクエスト数を入力するフィールドが表示され、新しい見積もりが表示されます。
6. 残りのディストリビューション設定を確認し、[ディストリビューションを作成する] を選択します。

ディストリビューションを作成すると、CloudFront は [セキュリティ] ダッシュボードを作成します。このダッシュボードを使用して、AWS WAF を無効化または有効化できます。AWS WAF をまだ有効にしていない場合、ダッシュボードのチャートとグラフは空白のままです。

既存のウェブ ACL を使用する

既存のウェブ ACL がある場合は、これを AWS WAF が提供する保護の代わりに使用できます。

既存の AWS WAF 設定を使用するには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。
2. 次のいずれかを行います。

   a. [ディストリビューションを作成] を選択し、「ディストリビューションを作成する」の手順を実行して、このトピックに戻ります。

   b. 既存の設定を選択し、[セキュリティ] タブを選択します。
3. [ウェブアプリケーションファイアウォール (WAF)] セクションで、[編集]、[セキュリティ保護を有効にする] の順に選択します。

4. [既存の WAF 設定を使用] を選択します。このオプションは、ウェブ ACL が設定されている場合にのみ表示されます。
5. [ウェブ ACL を選択] テーブルから既存のウェブ ACL を選択します。
6. 残りのディストリビューション設定を確認し、[ディストリビューションを作成する] を選択します。

Bot Control を有効にする

CloudFront ディストリビューションで AWS WAF を有効にすると、CloudFront コンソールのセキュリティダッシュボードで特定の時間範囲のボットリクエストを表示できます。ここで Bot Control を有効または無効にすることもできます。

Bot Control を有効にすると、料金が発生します。セキュリティダッシュボードには、コストの見積もりが表示されます。

Bot Control を有効にすると、セキュリティダッシュボードにボットのタイプおよびカテゴリ別にボットトラフィックが表示されます。Bot Control を無効にすると、リクエストサンプリングに基づいてボットトラフィックが表示されます。

Bot Control を有効にするには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。
2. ナビゲーションペインで、[ディストリビューション] を選択し、変更するディストリビューションを選択します。
3. [セキュリティ] タブを選択します。
4. [特定の時間範囲のボットリクエスト] セクションまでスクロールし、[Bot Control を有効にする] を選択します。
5. [Bot Control] ダイアログボックスの [設定] で、[一般的なボットの Bot Control を有効にする] チェックボックスをオンにします。
6. [Save changes] (変更の保存) をクリックします。

ボットカテゴリ別に保護を設定する

Bot Control を有効にすると、未確認の各ボットをボットカテゴリ別にどのように処理するかを設定できます。例えば、HTTP ライブラリボットを [モニタリングモード] に設定し、[チャレンジ] をリンクチェッカーに割り当てることができます。

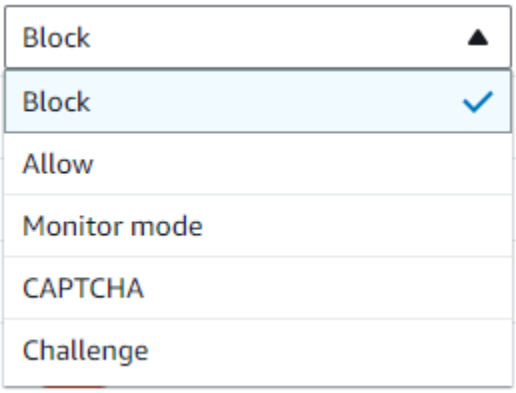
> **Note**
>
> 既知の検索エンジンのクローラーなど、一般的で検証可能として AWS に知られているボットは、ここで設定したアクションの対象にはなりません。Bot Control は、ボットを検証済みとしてマークする前に、検証済みのボットが請求するソースに由来することを確認します。

ボットカテゴリの保護を設定するには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。
2. ナビゲーションペインで、[ディストリビューション] を選択し、変更するディストリビューションを選択します。
3. [セキュリティ] タブを選択します。
4. [ボットカテゴリ別のリクエスト] グラフで、[未確認のボットアクション] 列のいずれかの項目にカーソルを合わせ、編集アイコンを選択します。

5. 結果のリストを開き、以下のいずれかを選択します。

   - ブロック
   - 許可
   - モニタリングモード
   - CAPTCHA
   - チャレンジ



6. リストの横にあるチェックマークをオンにして変更を確定します。

CloudFront セキュリティダッシュボードで AWS WAF セキュリティ保護を管理する

CloudFront はディストリビューションごとにセキュリティダッシュボードを作成します。ダッシュボードは、CloudFront コンソールで使用します。ダッシュボードにより、CloudFront と AWS WAF を 1 か所で使用して、ウェブアプリケーションの一般的なセキュリティ保護をモニタリングおよび管理できます。ダッシュボードは、以下のタスクとデータを提供します。

- セキュリティ設定 – AWS WAF 保護を有効または無効にし、WordPress 保護などのアプリケーション固有の保護を確認できます。
- セキュリティトレンド – 許可されたリクエストとブロックされたリクエスト、チャレンジリクエストとキャプチャリクエスト、上位の攻撃タイプが表示されます。トラフィックの比率と、その経時的な変化を確認できます。例えば、すべてのリクエストが 3% 増加し、許可されたリクエストが 14% 増加した場合、現在の期間でトラフィックを許可した割合が高かったことになります。
- ボットリクエスト – ボットからのトラフィック量、ボットタイプ (検証済みと未検証)、ボットタイプの割り当て率 (検証済みと未検証) が時間の経過に伴ってどのように変化するかを確認できます。Bot Control の有効化の詳細については、「Bot Control を有効にする」を参照してください。
- リクエストログ – ログデータは、セキュリティトレンドやボットリクエストに関する質問に回答するのに役立ちます。クエリを作成しなくてもログを検索し、集計グラフを表示できるため、フィルタリングされた一連のログが主に HTTP メソッド、IP アドレス、URI パス、または国のサブセットによって処理されているかどうかを判断できます。グラフの値にカーソルを合わせると、IP アドレスや国をブロックできます。詳細については、「AWS WAF のログを有効にする」を参照してください。
- 地理的制限の管理 — CloudFront と AWS WAF は、地理的制限機能を提供します。CloudFront は地理的制限を無料で提供しますが、CloudFront の地理的制限のメトリクスはセキュリティダッシュボードに表示されません。ブロックされた国のリクエストに関するリクエストメトリクスを表示するには、AWS WAF の地理的制限を使用する必要があります。この場合は、セキュリティダッシュボードの国バーにカーソルを合わせ、国をブロックします。詳細については、「CloudFront の地理的制限を使用する」を参照してください。
  - 以前に CloudFront コンソールの外部で国をブロックするようにカスタムの AWS WAF ルールを作成している場合、[ブロック] オプションは使用できないことがあります。

トピック

前提条件

CloudFront の [セキュリティ] ダッシュボードにセキュリティメトリクスを表示するには、AWS WAF を有効にする必要があります。AWS WAF を有効にしない場合、[セキュリティ] ダッシュボードでは AWS WAF を有効にするか、CloudFront の地理的制限を設定することだけができます。

AWS WAF の有効化の詳細については「ディストリビューションで AWS WAF を有効にする」をご覧ください。

AWS WAF のログを有効にする

AWS WAF のログデータは、特定のトラフィックパターンを切り分けるのに役立ちます。例えば、特定のトラフィックがどこから来ているのか、何をするのかをログで確認できます。

CloudWatch への AWS WAF のログ記録を有効にすると、CloudFront セキュリティダッシュボードは、CloudWatch ログのインサイトをクエリ、集計、表示します。セキュリティダッシュボードの使用には料金がかかりませんが、ダッシュボードからクエリしたログには CloudWatch の料金が適用されます。詳細については、「Amazon CloudWatch 料金表」をご覧ください。

ログを有効にするには

1. [1 か月あたりのリクエスト数] ボックスに予想されるリクエスト量を入力して、ログを有効にした場合のコストを見積もります。
2. [AWS WAF ログを有効にする] チェックボックスをオンにします。
3. [Enable（有効化）] を選択します。

CloudFront は CloudWatch ロググループを作成し、AWS WAF 設定を更新して CloudWatch へのログ記録を開始します。最初に使用する場合は、ログデータが表示されるまでに数分かかります。グラフの [リクエスト] セクションに、各リクエストが一覧表示されます。各リクエストの下にある棒グラフは、HTTP メソッド、上位の URI パス、上位の IP アドレス、上位の国ごとにデータを集計します。グラフは、パターンを確認するのに役立ちます。例えば、1 つの IP アドレスからの不釣り合いに多いリクエストや、以前はログに表示されていなかった国のデータを確認できます。国、ホストヘッダー、その他の属性に基づいてリクエストをフィルタリングすると、不要なトラフィックを見つ

けやすくなります。不要なトラフィックを特定したら、各リクエストやグラフ項目にカーソルを合わせ、IP アドレスまたは国をブロックします。

> **Note**
>
> 表示されるメトリクスは、ACL に基づいています。したがって、同じウェブ ACL を複数のディストリビューションに関連付けると、該当するディストリビューション用に処理された AWS WAF リクエストだけでなく、ウェブ ACL のすべてのメトリクスが表示されます。

レート制限の設定

レート制限は、セキュリティ保護を設定する際に受け取る可能性のある推奨事項の 1 つです。

CloudFront では、常にモニタリングモードでレート制限が有効になります。モニタリングモードを有効にすると、CloudFront は、[レート制限] フィールドで設定したレートを超過したかどうか、その頻度、および超過量を示すメトリクスをキャプチャします。

ディストリビューションを保存すると、CloudFront は [レート制限] フィールドの数に基づいてデータの収集を開始します。

レート制限の設定は、任意の CloudFront ディストリビューションで [セキュリティ] タブの [セキュリティ – ウェブアプリケーションファイアウォール (WAF)] セクションを使用して管理できます。

レート制限をセットアップするには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。
2. ナビゲーションペインで、[ディストリビューション] を選択し、変更するディストリビューションを選択します。
3. [セキュリティ] タブを選択します。
4. [ウェブアプリケーションファイアウォール (WAF)] セクションの [レート制限] の横で、[モニタリングモード] メッセージを選択して、収集したデータの詳細を示すダイアログを表示します。必要に応じてレート制限を変更できます。レートを微調整したら、ダイアログの [ブロックを有効にする] を選択してモニタリングモードを無効にできます。CloudFront は、指定されたレート制限を超えるリクエストのブロックを開始します。

AWS WAF セキュリティ保護を無効にする

ディストリビューションに AWS WAF セキュリティ保護が不要な場合は、この機能を CloudFront コンソールで無効にすることができます。

AWS WAF 保護を以前に有効にしていて、既存の WAF 設定 (ワンクリック保護とも呼ばれます) を選択しなかった場合、CloudFront は自動的にウェブ ACL を作成しました。この方法で作成されたウェブ ACL の場合、CloudFront コンソールはリソースの関連付けを解除し、ウェブ ACL を削除します。

ウェブ ACL の関連付けを解除することは、削除することとは異なります。関連付けを解除すると、ウェブ ACL はディストリビューションからは除外されますが、AWS アカウントからは削除されません。詳細については、「AWS WAF、AWS Firewall Manager、および AWS Shield Advanced デベロッパーガイド」の「ウェブ ACL と AWS リソースの関連付けまたは関連付け解除」を参照してください。

AWS WAF 保護を無効にし、ウェブ ACL とディストリビューションの関連付けを解除するには、次の手順を参照してください。

CloudFront で AWS WAF セキュリティ保護を無効にするには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。
2. ナビゲーションペインで、[ディストリビューション] を選択し、変更するディストリビューションを選択します。
3. [セキュリティ] タブを選択し、[編集] を選択します。
4. [ウェブアプリケーションファイアウォール (WAF)] セクションで、[AWS WAF 保護を無効にする] を選択します。
5. [Save changes] (変更の保存) をクリックします。

> **メモ**
>
> - AWS WAF セキュリティ保護を無効にした後で、ウェブ ACL を AWS アカウント から削除したい場合は、手動で削除できます。ウェブ ACL を削除する手順に従ってください。AWS WAF & Shield コンソールの [ウェブ ACL] ページで、[グローバル (CloudFront)] リストを選択してウェブ ACL を検索する必要があります。

- CloudFront コンソールからディストリビューションを削除すると、ワンクリック保護を選択している場合、CloudFront はウェブ ACL も削除しようとします。これはベストエフォートであり、常に保証されるわけではありません。詳細については、「ディストリビューションを削除する」を参照してください。

コンテンツへのセキュアなアクセスの設定とアクセスの制限

CloudFront には、コンテンツをセキュリティで保護するためのオプションが用意されています。CloudFront を使用してコンテンツへのアクセスを保護および制限する方法を次に示します。

- HTTPS 接続を設定する
- 特定の地理的な場所にいるユーザーがコンテンツにアクセスできないようにする
- CloudFront の署名付き URL または署名付き Cookie を使用してコンテンツにアクセスするようにユーザーに要求する
- 特定のコンテンツフィールドのフィールドレベルの暗号化を設定する
- AWS WAF を使用してコンテンツへのアクセスを管理する

トピック

CloudFront で HTTPS を使用する

ビューワーによる HTTPS の使用を必須とするように CloudFront を設定して、CloudFront がビューワーと通信するときの接続を暗号化することができます。また、オリジンに HTTPS を使用するように CloudFront を設定して、CloudFront がオリジンと通信するときの接続を暗号化することも可能です。

ビューワーとの通信とオリジンとの通信の両方で HTTPS を必須とするように CloudFront を設定する場合、CloudFront がオブジェクトのリクエストを受け取るときに以下が行われます。

1. ビューワーが HTTPS リクエストを CloudFront に送信します。ここで、ビューワーと CloudFront 間で何らかの SSL/TLS ネゴシエーションが実行されます。最終的に、ビューワーはリクエストを暗号化形式で送信します。

2. CloudFront のエッジロケーションにキャッシュされたレスポンスがある場合、CloudFront はレスポンスを暗号化してビューワーに返し、ビューワーがそれを復号化します。
3. CloudFront のエッジロケーションにキャッシュされたレスポンスがない場合、CloudFront はオリジンとの SSL/TLS ネゴシエーションを実行し、ネゴシエーションが完了すると、暗号化された形式でリクエストをオリジンに転送します。
4. オリジンは、リクエストを復号化して処理し (レスポンスを生成)、レスポンスを暗号化してから、そのレスポンスを CloudFront に返します。
5. CloudFront はレスポンスを復号化してから再度暗号化し、それをビューワーに転送します。また、CloudFront はエッジロケーションにレスポンスをキャッシュして、次回リクエストされたときに使用できるようにします。
6. ビューワーは応答の暗号化を解除します。

オリジンが Amazon S3 バケット、MediaStore、または HTTP/S サーバーなどのカスタムオリジンのどちらであるかに関係なく、プロセスは基本的に同じように機能します。

> **Note**
>
> SSL の再ネゴシエーション型攻撃を阻止するために、CloudFront はビューワーリクエストとオリジンリクエストの再ネゴシエーションをサポートしていません。

ビューワーと CloudFront との間、また CloudFront とオリジンとの間で HTTPS を要求する方法については、次のトピックを参照してください。

トピック

- ビューワーと CloudFront の間の通信に HTTPS を要求する
- CloudFront とカスタムオリジンの間の通信に HTTPS を要求する
- CloudFront と Amazon S3 オリジンの間の通信に HTTPS を要求する
- ビューワーと CloudFront との間でサポートされているプロトコルと暗号
- CloudFront とオリジンとの間でサポートされているプロトコルと暗号

ビューワーと CloudFront の間の通信に HTTPS を要求する

CloudFront ディストリビューション内で 1 つ以上のキャッシュ動作を設定して、ビューワーと CloudFront との通信で HTTPS を必須にできます。また、1 つ以上のキャッシュ動作で HTTP と

HTTPS の両方を許可するように設定して、CloudFront における一部のオブジェクトで HTTPS を必須にすることもできます。設定手順はオブジェクト URL 内で使用しているドメイン名によって異なります。

- CloudFront がディストリビューションに割り当てたドメイン名を使用している場合 (d111111abcdef8.cloudfront.net など)、1 つ以上のキャッシュ動作の [ビューワープロトコルポリシー] 設定を変更して、HTTPS 通信を必須にします。この設定で、CloudFront は SSL/TLS 証明書を提供します。

  CloudFront コンソールを使用して [ビューワープロトコルポリシー] の値を変更するには、このセクションで後述する手順を参照してください。

  CloudFront API を使用して `ViewerProtocolPolicy` 要素の値を変更する方法については、Amazon CloudFront API リファレンスの「UpdateDistribution」を参照してください。
- 独自のドメイン名 (example.com など) を使用している場合、CloudFront のいくつかの設定を変更する必要があります。また、AWS Certificate Manager (ACM) が提供する SSL/TLS 証明書を使用するか、サードパーティー認証機関からの証明書を ACM または IAM 証明書ストアにインポートする必要があります。詳細については、「代替ドメイン名と HTTPS を使用する」を参照してください。

> **Note**
>
> ビューワーが CloudFront から取得するオブジェクトに関して、CloudFront がそのオブジェクトをオリジンから取得した段階で暗号化されていることを保証するには、CloudFront とオリジンとの間で必ず HTTPS を使用します。最近になって CloudFront とオリジンとの間で HTTP から HTTPS に変更した場合、CloudFront エッジロケーションのオブジェクトを無効にすることをお勧めします。CloudFront は、ビューワーが使用しているプロトコル (HTTP または HTTPS) について、CloudFront がオブジェクトを取得するのに使用したプロトコルと一致するかどうかに関係なく、オブジェクトをビューワーに返します。ディストリビューション内のオブジェクトの削除または置き換えの詳細については、「CloudFront が配信するコンテンツを追加、削除、または置き換える」を参照してください。

ビューワーに HTTPS を要求する

1 つ以上のキャッシュ動作でビューワーと CloudFront との間で HTTPS を必須にするには、次の手順を実行します。

ビューワーと CloudFront の間で HTTPS が必須になるよう CloudFront を設定するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. CloudFront コンソールの上部のペインで、更新するディストリビューションの ID を選択します。
3. [ビヘイビア] タブで、更新するキャッシュビヘイビアを選択し、[編集] を選択します。
4. [ビューワープロトコルポリシー] で、次のいずれかの値を指定します。

   Redirect HTTP to HTTPS

   ビューワーは両方のプロトコルを使用できます。HTTP GET および HEAD リクエストは自動的に HTTPS リクエストにリダイレクトされます。CloudFront は新しい HTTPS URL とともに HTTP ステータスコード 301 (Moved Permanently) を返します。ビューワーはこの HTTPS URL を使用して CloudFront にリクエストを再送信します。

   > **Important**
   >
   > POST、PUT、DELETE、OPTIONS、PATCH を HTTP 経由で HTTP から HTTPS キャッシュ動作にして HTTP 1.1 以降のリクエストプロトコルバージョンで送信すると、CloudFront はそのリクエストを HTTP ステータスコード 307 (Temporary Redirect) で HTTPS の場所にリダイレクトします。これはリクエストが同じメソッドと本文ペイロードを使用して新しい場所に再度送信されることを保証するものです。
   >
   > POST、PUT、DELETE、OPTIONS、PATCH のリクエストを HTTP 経由で HTTPS キャッシュ動作に HTTP 1.1 以前のリクエストプロトコルバージョンで送信すると、CloudFront は HTTP ステータスコード 403 (Forbidden) を返します。

   ビューワーが作成した HTTP リクエストが HTTPS リクエストにリダイレクトされた場合、CloudFront では両方のリクエストに対する課金が発生します。HTTP リクエストの場合は、リクエストの料金と、CloudFront がビューワーに返すヘッダーの料金のみが課金されます。HTTPS リクエストの場合、リクエストの料金と、オリジンが返すヘッダーとオブジェクトの料金が課金されます。

HTTPS のみ

ビューワーは、HTTPS を使用している場合にのみ、コンテンツにアクセスできます。ビューワーが HTTPS リクエストではなく HTTP リクエストを送信した場合、CloudFront は HTTP ステータスコード 403 (Forbidden) を返し、オブジェクトは返しません。

5. [Save changes] (変更の保存) をクリックします。
6. ビューワーと CloudFront との間で HTTPS を必須にする追加のキャッシュ動作ごとに、ステップ 3 から 5 を繰り返します。
7. 本番環境で更新された情報を使用する前に、次を確認してください。

   - ビューワーに HTTPS の使用が必要とされるリクエストにのみ、各キャッシュ動作のパスパターンが適用されている。
   - CloudFront が評価する順番にキャッシュ動作がリストされている。詳細については、「パスパターン」を参照してください。
   - キャッシュ動作は、リクエストを正しいオリジンにルーティングします。

CloudFront とカスタムオリジンの間の通信に HTTPS を要求する

CloudFront とカスタムオリジン間の通信で HTTPS を必須にすることができます。

> **Note**
>
> オリジンがウェブサイトエンドポイントとして設定されている Amazon S3 バケットである場合、Amazon S3 がウェブサイトエンドポイントに対して HTTPS をサポートしていないため、オリジンに HTTPS を使用するように CloudFront を設定することはできません。

CloudFront とオリジン間で HTTPS を必須にするには、このトピックの手順に従って次を実行します。

1. ディストリビューションで、[Origin Protocol Policy] (オリジンプロトコルポリシー) 設定を変更します。
2. オリジンサーバーに SSL/TLS 証明書をインストールします (Amazon S3 オリジン、または特定のその他の AWS オリジンを使用する場合は必要ありません)。

トピック

- カスタムオリジンに HTTPS を要求する
- カスタムオリジンに SSL/TLS 証明書をインストールする

カスタムオリジンに HTTPS を要求する

次の手順では、Elastic Load Balancing ロードバランサー、Amazon EC2 インスタンス、または別のカスタムオリジンとの通信で HTTPS を使用するよう CloudFront を設定する方法について説明します。CloudFront API を使用してディストリビューションを更新する方法については、Amazon CloudFront API リファレンスの「UpdateDistribution」を参照してください。

CloudFront とカスタムオリジンの間で HTTPS を必須にするよう CloudFront を設定するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. CloudFront コンソールの上部のペインで、更新するディストリビューションの ID を選択します。
3. [ビヘイビア]] タブで、更新するオリジンを選択し、[編集] を選択します。
4. 次の設定を更新します。

   オリジンプロトコルポリシー

   ディストリビューションの該当するオリジンで、[Origin Protocol Policy] を変更します。

   - [HTTPS Only (HTTPS のみ)] - CloudFront は HTTPS のみを使ってカスタムオリジンと通信します。
   - [Match Viewer (ビューワーに合わせる)] - CloudFront は、ビューワーのリクエストのプロトコルに応じて HTTP または HTTPS を使用し、カスタムオリジンと通信します。例えば、[オリジンプロトコルポリシー] の [Match Viewer (ビューワーに合わせる)] を選択し、ビューワーで HTTPS を使用して CloudFront からオブジェクトをリクエストする場合は、CloudFront でも HTTPS を使用してリクエストをオリジンに転送します。

     [Viewer Protocol Policy] で [Redirect HTTP to HTTPS] または [HTTPS Only] を指定する場合は、[Match Viewer] のみを選択します。

     ビューワーが HTTP と HTTPS の両方のプロトコルを使用してリクエストを行った場合も、CloudFront がオブジェクトをキャッシュするのは 1 回だけです。

オリジン SSL プロトコル

ディストリビューションの該当するオリジンで [Origin SSL Protocols] を選択します。SSLv3 プロトコルは安全性が低いため、オリジンが TLSv1 以降をサポートしていない場合にのみ SSLv3 を選択することをお勧めします。TLSv1 ハンドシェイクは、SSLv3 との下位互換性と上位互換性の両方がありますが、TLSv1.1 以降は対象外です。SSLv3 を選択すると、CloudFront は SSLv3 ハンドシェイクリクエストのみを送信します。

5. [Save changes] (変更の保存) をクリックします。
6. CloudFront とカスタムオリジンとの間で HTTPS を必須にする追加のオリジンごとに、ステップ 3 から 5 を繰り返します。
7. 本番環境で更新された情報を使用する前に、次を確認してください。

   - ビューワーに HTTPS の使用が必要とされるリクエストにのみ、各キャッシュ動作のパスパターンが適用されている。
   - CloudFront が評価する順番にキャッシュ動作がリストされている。詳細については、「パスパターン」を参照してください。
   - キャッシュ動作は、[Origin Protocol Policy] を変更したオリジンにリクエストをルーティングします。

カスタムオリジンに SSL/TLS 証明書をインストールする

SSL/TLS 証明書は、カスタムオリジンの次のソースから使用できます。

- オリジンが Elastic Load Balancing ロードバランサーである場合は、AWS Certificate Manager (ACM) で提供された証明書を使用できます。信頼されたサードパーティー認証機関が署名して ACM にインポートされた証明書を使用することもできます。
- Elastic Load Balancing ロードバランサー以外のオリジンの場合、信頼されたサードパーティー認証機関 (CA) (Comodo、DigiCert、Symantec など) によって署名された証明書を使用する必要があります。

オリジンから返される証明書には、次のいずれかのドメイン名が含まれている必要があります。

- オリジンのオリジンドメインフィールド (CloudFront API の `DomainName` フィールド) のドメイン名。
- キャッシュ動作が `Host` ヘッダーをオリジンに転送するように設定されている場合は、`Host` ヘッダーのドメイン名。

CloudFront が HTTPS を使用してオリジンと通信するとき、CloudFront は証明書が信頼された認証機関によって発行されたものであることを確認します。CloudFront は Mozilla と同じ認証機関をサポートしています。最新のリストは、「Mozilla に付属する CA 証明書一覧」を参照してください。CloudFront とオリジンとの間の HTTPS 通信に自己署名証明書を使用することはできません。

> **Important**
>
> 失効した証明書、無効な証明書、または自己署名証明書をオリジンサーバーが返したり、間違った順番の証明書チェーンを返したりした場合、CloudFront は TCP 接続を中断し、HTTP ステータスコード 502 (Bad Gateway) を返して、`X-Cache` ヘッダーを `Error from cloudfront` に設定します。中間証明書を含む、証明書チェーンが完全でない場合も、CloudFront は TCP 接続を中断します。

CloudFront と Amazon S3 オリジンの間の通信に HTTPS を要求する

オリジンが Amazon S3 バケットの場合、CloudFront との通信に HTTPS を使用するためのオプションは、バケットの使用方法によって異なります。Amazon S3 バケットがウェブサイトエンドポイントとして設定されている場合、オリジンとの通信に HTTPS を使用するように CloudFront を設定することはできません。Amazon S3 はその設定で HTTPS 接続をサポートしていないためです。

オリジンが HTTPS 通信をサポートする Amazon S3 バケットの場合、CloudFront はビューワーがリクエストを送信するのに使ったプロトコルを必ず使用して S3 にリクエストを転送します。プロトコル (カスタムオリジンのみ) 設定のデフォルト設定は [Match Viewer (ビューワーに合わせる)] で、変更できません。

CloudFront と Amazon S3 との間の通信で HTTPS を必須にする場合、[ビューワープロトコルポリシー] の値を [Redirect HTTP to HTTPS (HTTP から HTTPS へのリダイレクト)] または [HTTPS Only (HTTPS のみ)] に変更する必要があります。このセクションで後述する手順では、CloudFront コンソールを使用して [ビューワープロトコルポリシー] を変更する方法について説明します。CloudFront API を使用してディストリビューションの `ViewerProtocolPolicy` 要素を更新する方法については、Amazon CloudFront API リファレンスの「UpdateDistribution」を参照してください。

HTTPS 通信をサポートする Amazon S3 バケットで HTTPS を使用する場合、Amazon S3 では SSL/TLS 証明書を使用できるため、この通信を使用する必要はありません。

Amazon S3 オリジンに HTTPS を要求する

次の手順は、Amazon S3 オリジンに HTTPS を要求するように CloudFront を設定する方法を示しています。

Amazon S3 オリジン接続時の HTTPS を必須にするよう CloudFront を設定するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. CloudFront コンソールの上部のペインで、更新するディストリビューションの ID を選択します。
3. [Behaviors] タブで、更新するキャッシュ動作を選択した後、[Edit] を選択します。
4. [Viewer Protocol Policy] として次のいずれかの値を指定します。

   Redirect HTTP to HTTPS

   ビューワーは両方のプロトコルを使用できますが、HTTP リクエストは自動的に HTTPS リクエストにリダイレクトされます。CloudFront は新しい HTTPS URL とともに HTTP ステータスコード 301 (Moved Permanently) を返します。ビューワーはこの HTTPS URL を使用して CloudFront にリクエストを再送信します。

   > **Important**
   >
   > CloudFront は、HTTP から HTTPS に DELETE、OPTIONS、PATCH、POST、または PUT リクエストをリダイレクトしません。HTTPS にリダイレクトするようにキャッシュ動作を設定した場合、CloudFront は HTTP ステータスコード 403 (Forbidden) を使用してそのキャッシュ動作の HTTP DELETE、OPTIONS、PATCH、POST、または PUT リクエストに応答します。

   ビューワーが作成した HTTP リクエストが HTTPS リクエストにリダイレクトされた場合、CloudFront では両方のリクエストに対する課金が発生します。HTTP リクエストの場合は、リクエストの料金と、CloudFront がビューワーに返すヘッダーの料金のみが課金されます。HTTPS リクエストの場合、リクエストの料金と、オリジンが返すヘッダーとオブジェクトの料金が課金されます。

HTTPS Only

ビューワーは、HTTPS を使用している場合にのみ、コンテンツにアクセスできます。ビューワーが HTTPS リクエストではなく HTTP リクエストを送信した場合、CloudFront は HTTP ステータスコード 403 (Forbidden) を返し、オブジェクトは返しません。

5. [Yes, Edit (はい、編集します)] を選択します。
6. ビューワーと CloudFront、および CloudFront と S3 との間で HTTPS を必須にする追加のキャッシュ動作ごとに、ステップ 3 から 5 を繰り返します。
7. 本番環境で更新された情報を使用する前に、次を確認してください。

   - ビューワーに HTTPS の使用が必要とされるリクエストにのみ、各キャッシュ動作のパスパターンが適用されている。
   - CloudFront が評価する順番にキャッシュ動作がリストされている。詳細については、「パスパターン」を参照してください。
   - キャッシュ動作は、リクエストを正しいオリジンにルーティングします。

ビューワーと CloudFront との間でサポートされているプロトコルと暗号

ビューワーと CloudFront ディストリビューションとの間で HTTPS が必要である場合は、セキュリティポリシーを選択する必要があります。これにより、次の設定が決定されます。

- CloudFront でビューワーとの通信に使用する最小の SSL/TLS プロトコル。
- ビューワーとの通信を暗号化するために CloudFront が使用できる暗号。

セキュリティポリシーを選択するには、セキュリティポリシー (SSL/TLS の最小バージョン) に該当する値を指定します。以下の表は、各セキュリティポリシーで CloudFront が使用できるプロトコルと暗号化方式の一覧です。

ビューワーは、CloudFront との HTTPS 接続を確立するために、これらの暗号のうち少なくとも 1 つをサポートする必要があります。CloudFront は、ビューワーがサポートする暗号化方式から一覧順で暗号化方式を選択します。「OpenSSL、s2n、および RFC の暗号名」も参照してください。

	セキュリティポリシー						
	SSLv3	TLSv1	TLSv1_26	TLSv1.1_016	TLSv1.2_018	TLSv1.2_019	TLSv1.2_2021
サポートされている SSL/TLS プロトコル							
TLSv1.3	◆	◆	◆	◆	◆	◆	◆
TLSv1.2	◆	◆	◆	◆	◆	◆	◆
TLSv1.1	◆	◆	◆	◆			
TLSv1	◆	◆	◆				
SSLv3	◆						
サポートされている TLSv1.3 暗号							
TLS_AES_128_GCM_SHA256	◆	◆	◆	◆	◆	◆	◆
TLS_AES_256_GCM_SHA384	◆	◆	◆	◆	◆	◆	◆
TLS_CHACHA20_POLY1305_SHA256	◆	◆	◆	◆	◆	◆	◆
サポートされる ECDSA 暗号							
ECDHE-ECDSA-AES128-GCM-SHA256	◆	◆	◆	◆	◆	◆	◆
ECDHE-ECDSA-AES128-SHA256	◆	◆	◆	◆	◆	◆	
ECDHE-ECDSA-AES128-SHA	◆	◆	◆	◆			

	セキュリティポリシー						
	SSLv3	TLSv1	TLSv1_26	TLSv1.1_016	TLSv1.2_018	TLSv1.2_019	TLSv1.2_2021
ECDHE-ECDSA-AES256-GCM-SHA384	♦	♦	♦	♦	♦	♦	♦
ECDHE-ECDSA-CHACHA20-POLY1305	♦	♦	♦	♦	♦	♦	♦
ECDHE-ECDSA-AES256-SHA384	♦	♦	♦	♦	♦	♦	
ECDHE-ECDSA-AES256-SHA	♦	♦	♦	♦			
サポートされる RSA 暗号							
ECDHE-RSA-AES128-GCM-SHA256	♦	♦	♦	♦	♦	♦	♦
ECDHE-RSA-AES128-SHA256	♦	♦	♦	♦	♦	♦	
ECDHE-RSA-AES128-SHA	♦	♦	♦	♦			
ECDHE-RSA-AES256-GCM-SHA384	♦	♦	♦	♦	♦	♦	♦
ECDHE-RSA-CHACHA20-POLY1305	♦	♦	♦	♦	♦	♦	♦
ECDHE-RSA-AES256-SHA384	♦	♦	♦	♦	♦	♦	
ECDHE-RSA-AES256-SHA	♦	♦	♦	♦			

	セキュリティポリシー						
	SSLv3	TLSv1	TLSv1_26	TLSv1.1_016	TLSv1.2_018	TLSv1.2_019	TLSv1.2_2021
AES128-GCM-SHA256	◆	◆	◆	◆	◆		
AES256-GCM-SHA384	◆	◆	◆	◆	◆		
AES128-SHA256	◆	◆	◆	◆	◆		
AES256-SHA	◆	◆	◆	◆			
AES128-SHA	◆	◆	◆	◆			
DES-CBC3-SHA	◆	◆					
RC4-MD5	◆						

OpenSSL、s2n、および RFC の暗号名

OpenSSL と s2n では、TLS 標準で使用されている暗号名 (RFC 2246、RFC 4346、RFC 5246、RFC 8446) とは異なる暗号名が使用されます。以下の表では、各暗号化方式の OpenSSL 名と s2n 名から RFC 名までを示しています。

楕円曲線キー交換アルゴリズムを使用した暗号について、CloudFront は次の楕円曲線をサポートします。

- prime256v1
- secp384r1
- X25519

OpenSSL および s2n の暗号名	RFC の暗号名
サポートされている TLSv1.3 暗号	
TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256

OpenSSL および s2n の暗号名	RFC の暗号名
TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256
サポートされる ECDSA 暗号	
ECDHE-ECDSA-AES128- GCM-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
ECDHE-ECDSA-AES128-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
ECDHE-ECDSA-AES128-SHA	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
ECDHE-ECDSA-AES256- GCM-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
ECDHE-ECDSA-CHACHA20-POLY1305	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
ECDHE-ECDSA-AES256-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
ECDHE-ECDSA-AES256-SHA	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
サポートされる RSA 暗号	
ECDHE-RSA-AES128- GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

OpenSSL および s2n の暗号名	RFC の暗号名
ECDHE-RSA-AES256- GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_ GCM_SHA384
ECDHE-RSA-CHACHA20-POLY1305	TLS_ECDHE_RSA_WITH_CHACHA20 _POLY1305_SHA256
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_ CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_ CBC_SHA
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
AES256-GCM-SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384
AES128-SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA
RC4-MD5	TLS_RSA_WITH_RC4_128_MD5

ビューワーと CloudFront 間でサポートされている署名スキーム

CloudFront では、ビューワーと CloudFront 間の接続について、次の署名スキームがサポートされています。

- TLS_SIGNATURE_SCHEME_RSA_PSS_PSS_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PSS_PSS_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PSS_PSS_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PSS_RSAE_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PSS_RSAE_SHA384

- TLS_SIGNATURE_SCHEME_RSA_PSS_RSAE_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA224
- TLS_SIGNATURE_SCHEME_ECDSA_SHA256
- TLS_SIGNATURE_SCHEME_ECDSA_SHA384
- TLS_SIGNATURE_SCHEME_ECDSA_SHA512
- TLS_SIGNATURE_SCHEME_ECDSA_SHA224
- TLS_SIGNATURE_SCHEME_ECDSA_SECP256R1_SHA256
- TLS_SIGNATURE_SCHEME_ECDSA_SECP384R1_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA1
- TLS_SIGNATURE_SCHEME_ECDSA_SHA1

CloudFront とオリジンとの間でサポートされているプロトコルと暗号

[CloudFront とオリジンとの間で HTTPS を必須にする] を選択する場合、安全な接続にどの SSL/TLS プロトコルを許可するかを決定し、次の表に示す ECDSA あるいは RSA 暗号のいずれかを使用して CloudFront からオリジンに接続できます。CloudFront がオリジンに対する HTTPS 接続を確立するには、オリジンがこれらの暗号のうち少なくとも 1 つをサポートしている必要があります。

OpenSSL と s2n では、TLS 標準で使用されている暗号名 (RFC 2246、RFC 4346、RFC 5246、RFC 8446) とは異なる暗号名が使用されます。次の表には、各暗号化方式の OpenSSL 名、s2n 名、および RFC 名などが示されています。

楕円曲線キー交換アルゴリズムを使用した暗号について、CloudFront は次の楕円曲線をサポートします。

- prime256v1
- secp384r1
- X25519

OpenSSL および s2n の暗号名	RFC の暗号名
サポートされる ECDSA 暗号	
ECDHE-ECDSA-AES256- GCM-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
ECDHE-ECDSA-AES256-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
ECDHE-ECDSA-AES256-SHA	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
ECDHE-ECDSA-AES128- GCM-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
ECDHE-ECDSA-AES128-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
ECDHE-ECDSA-AES128-SHA	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
サポートされる RSA 暗号	
ECDHE-RSA-AES256- GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
ECDHE-RSA-AES128- GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

OpenSSL および s2n の暗号名	RFC の暗号名
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA
RC4-MD5	TLS_RSA_WITH_RC4_128_MD5

CloudFront とオリジン間でサポートされている署名スキーム

CloudFront では、CloudFront とオリジン間の接続について、次の署名スキームがサポートされています。

- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA224
- TLS_SIGNATURE_SCHEME_ECDSA_SHA256
- TLS_SIGNATURE_SCHEME_ECDSA_SHA384
- TLS_SIGNATURE_SCHEME_ECDSA_SHA512
- TLS_SIGNATURE_SCHEME_ECDSA_SHA224
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA1
- TLS_SIGNATURE_SCHEME_ECDSA_SHA1

代替ドメイン名と HTTPS を使用する

ファイルの URL で独自ドメイン名を使用する場合 (例: `https://www.example.com/image.jpg`)、およびビューワーが HTTPS を使用する場合、次のトピックのステップを完了する必要があります。(URL でデフォルトの CloudFront ディストリビューションのドメイン名、たとえば `https://d111111abcdef8.cloudfront.net/image.jpg` を使用する場合は、代わりに「ビューワーと CloudFront の間の通信に HTTPS を要求する」のガイダンスに従います)。

> **Important**
>
> 証明書をディストリビューションに追加すると、CloudFront によりすべてのエッジロケーションに証明書が直ちに伝達されます。新しいエッジロケーションが使用可能になると、CloudFront は証明書をそのロケーションにも伝達します。CloudFront から証明書がどのエッジロケーションに伝達されるかを制限することはできません。

トピック

CloudFront で HTTPS リクエストを処理する方法を選択する

ビューワーに HTTPS とファイルの代替ドメイン名を使用させる場合、CloudFront サーバーが HTTPS リクエストを処理する方法に関する次のいずれかのオプションを選択する必要があります。

- Server Name Indication (SNI) を使用する - 推奨
- 各エッジロケーションの専用 IP アドレスを使用する

このセクションでは各オプションの仕組みについて説明します。

SNI を使用して HTTPS リクエストを処理する (ほとんどのクライアントで有効)

Server Name Indication (SNI) は、2010 年以降にリリースされたブラウザとクライアントでサポートされている TLS プロトコルを拡張したものです。SNI を使用して HTTPS リクエストに対応するように CloudFront を設定した場合、CloudFront は代替ドメイン名を各エッジロケーションの IP ア

ドレスと関連付けます。ビューワーがコンテンツに対して HTTPS リクエストを送信すると、DNS は、正しいエッジロケーションの IP アドレスにリクエストをルーティングします。ドメイン名の IP アドレスが SSL/TLS ハンドシェイクネゴシエーション中に決定されます。IP アドレスはディストリビューション専用にはなりません。

SSL/TLS ネゴシエーションは、HTTPS 接続を確立する処理の早い段階で実行されます。リクエストがどのドメイン向けかをすぐに決定できない場合、CloudFront は接続を中断します。SNI をサポートするビューワーがコンテンツに対して HTTPS リクエストを送信すると、次のようになります。

1. ビューワーはリクエスト URL からドメイン名を自動的に取得し、TLS クライアントの hello メッセージの SNI 拡張に追加します。
2. CloudFront は、TLS クライアント hello を受け取ると、SNI 拡張のドメイン名を使用して、一致する CloudFront ディストリビューションを検索し、関連する TLS 証明書を返します。
3. ビューワーと CloudFront は、SSL/TLS ネゴシエーションを実行します。
4. CloudFront はリクエストされたコンテンツをビューワーに返します。

現在 SNI をサポートするブラウザの一覧については、Wikipedia の Server Name Indication の項目を参照してください。

SNI を使用したくても、ユーザーのブラウザの一部が SNI をサポートしていない場合は、選択肢がいくつかあります。

- SNI ではなく専用 IP アドレスを使用して HTTPS リクエストを供給するように CloudFront を構成します。詳細については、「専用 IP アドレスを使用して HTTPS リクエストを処理する (すべてのクライアントで有効)」を参照してください。
- カスタム証明書の代わりに CloudFront SSL/TLS 証明書を使用してください。この場合、ファイルの URL でディストリビューションの CloudFront ドメイン名を使用する必要があります (たとえば、`https://d111111abcdef8.cloudfront.net/logo.png`)。

  デフォルトの CloudFront 証明書を使用する場合、ビューワーは SSL プロトコル TLSv1 以降をサポートする必要があります。CloudFront はデフォルトの CloudFront 証明書では SSLv3 をサポートしません。

  また、CloudFront で使用する SSL/TLS 証明書を独自証明書からデフォルトの CloudFront 証明書に変更する必要もあります。

  - ディストリビューションを使用してコンテンツを配信したことがない場合は、単に構成を変更できます。詳細については、「ディストリビューションを更新する」を参照してください。

  - ディストリビューションを使用してコンテンツを配信していた場合は、新しい CloudFront ディストリビューションを作成し、コンテンツが使用できない時間を減らすかゼロにするために、ファイルの URL を変更する必要があります。詳細については、「カスタム SSL/TLS 証明書からデフォルト CloudFront 証明書に戻す」を参照してください。
- ユーザーが使用するブラウザを管理できる場合は、SNI をサポートするブラウザにアップグレードしてもらいます。
- HTTPS の代わりに HTTP を使用します。

専用 IP アドレスを使用して HTTPS リクエストを処理する (すべてのクライアントで有効)

Server Name Indication (SNI) は、リクエストをドメインと関連付ける 1 つの方法です。専用 IP アドレスを使用する方法もあります。2010 年以降にリリースされたブラウザまたはクライアントにアップグレードできないユーザーが存在する場合は、専用 IP アドレスを使用して HTTPS リクエストに対応できます。現在 SNI をサポートするブラウザの一覧については、Wikipedia の Server Name Indication の項目を参照してください。

> **Important**
>
> 専用 IP アドレスを使用して HTTPS リクエストを供給するように CloudFront を構成した場合、追加の月額料金が発生します。課金は、ディストリビューションに SSL/TLS 証明書を関連付けて、ディストリビューションを有効にした時点から開始されます。CloudFront の料金の詳細については、「Amazon CloudFront の料金」を参照してください。また、Using the Same Certificate for Multiple CloudFront Distributions も参照してください。

専用 IP アドレスを使用して HTTPS リクエストに対応するように CloudFront を設定すると、CloudFront は証明書を各 CloudFront エッジロケーションの専用 IP アドレスと関連付けます。ビューワーがコンテンツに HTTPS リクエストを送信すると、次のようになります。

1. DNS は、該当するエッジロケーション内のディストリビューションの IP アドレスにリクエストをルーティングします。
2. クライアントリクエストの `ClientHello` メッセージに SNI 拡張が含まれている場合、CloudFront は、その SNI に関連付けられているディストリビューションを検索します。
   - 一致するものがあれば、CloudFront は SSL/TLS 証明書を使用してリクエストに応答します。

- 一致するものがない場合、CloudFront は代わりに IP アドレスを使用してディストリビューションを特定し、どの SSL/TLS 証明書をビューワーに返すかを決定します。

3. ビューワーと CloudFront は SSL/TLS 証明書を使って SSL/TLS ネゴシエーションを実行します。
4. CloudFront はリクエストされたコンテンツをビューワーに返します。

この方法は、ユーザーが使用するブラウザやその他のビューワーを問わず、あらゆる HTTPS リクエストで機能します。

3 つ以上の SSL/TLS 専用 IP 証明書を使用するためのアクセス許可をリクエストする

CloudFront に 3 つ以上の SSL/TLS 専用 IP 証明書を永続的に関連付けるための許可が必要な場合は、以下の手順を実行します。HTTPS リクエストの詳細については、「CloudFront で HTTPS リクエストを処理する方法を選択する」を参照してください。

> **Note**
>
> CloudFront ディストリビューションに 3 つ以上の専用 IP 証明書を使用するための手順は以下の通りです。デフォルト値は 2 です。ディストリビューションには SSL 証明書を複数バインドできないのでご注意ください。
> CloudFront ディストリビューションに関連付けることができる SSL/TLS 証明書は、一度に 1 つのみです。この数は、すべての CloudFront ディストリビューションで使用できる専用 IP SSL 証明書の合計数です。

CloudFront ディストリビューションに 3 つ以上の証明書を使用するための許可をリクエストするには

1. サポートセンターにアクセスし、サポートケースを作成します。
2. 使用するためのアクセス権限が必要な証明書の数と状況をリクエストに記載してください。できる限り早くアカウントを更新します。
3. 次の手順に進みます。

CloudFront で SSL/TLS 証明書を使用するための要件

このトピックでは、SSL/TLS 証明書の要件について説明します。これらの要件は、特に注記がなければ、以下の両方の証明書に適用されます。

- ビューワーと CloudFront との間で HTTPS を使用するための証明書
- CloudFront とオリジンとの間で HTTPS を使用するための証明書

トピック

証明書の発行者

AWS Certificate Manager (ACM) で発行された証明書を使用することをお勧めします。ACM からの証明書の取得の詳細については、AWS Certificate Manager ユーザーガイドを参照してください。CloudFront で ACM 証明書を使用するには、米国東部 (バージニア北部) リージョン (us-east-1) の証明書をリクエスト (またはインポート) していることを確認します。

CloudFront は Mozilla と同じ認定権限 (CA) をサポートしているため、ACM を使用しない場合は、「Mozilla に付属する CA 証明書一覧」の CA が発行する証明書を使用します。証明書の取得とインストール方法については、使用している HTTP サーバーソフトウェアのドキュメントおよび CA のドキュメントを参照してください。

AWS Certificate Manager の場合は AWS リージョン

AWS Certificate Manager (ACM) の証明書を使用して、ビューワーと CloudFront との間で HTTPS を必須にする場合は、米国東部 (バージニア北部) リージョン (us-east-1) の証明書をリクエスト (またはインポート) していることを確認します。

CloudFront とオリジンとの間で HTTPS を必須にする場合、オリジンとして Elastic Load Balancing のロードバランサーを使用していれば、任意の AWS リージョン で証明書をリクエストまたはインポートできます。

証明書の形式

公開証明書は X.509 PEM 形式で作成されている必要があります。これは、AWS Certificate Manager を使用する場合のデフォルトの形式です。

中間証明書

サードパーティー認定権限 (CA) を使用している場合、.pem ファイルには、ドメインの証明書の署名者である CA の証明書から始めて、証明書チェーン内のすべての中間証明書を含めます。通常は、適切なチェーン順で中間証明書とルート証明書を並べたファイルが CA のウェブサイトに用意されています。

> **Important**
>
> ルート証明書、信頼パス内に存在しない中間証明書、CA の公開キー証明書は含めないでください。

例を示します。

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 2
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Intermediate certificate 1
-----END CERTIFICATE-----
```

キーのタイプ

CloudFront は RSA および ECDSA パブリック (公開) とプライベートのキーペアをサポートします。

CloudFront は、RSA 証明書と ECDSA 証明書を使用して、ビューワーとオリジンの両方への HTTPS 接続をサポートします。AWS Certificate Manager (ACM) では、RSA または ECDSA 証明書をリクエストしてインポートし、CloudFront ディストリビューションに関連付けることができます。

HTTPS 接続でネゴシエートできる CloudFront でサポートされている RSA および ECDSA 暗号のリストについては、「the section called “ビューワーと CloudFront との間でサポートされているプロトコルと暗号”」および「the section called “CloudFront とオリジンとの間でサポートされているプロトコルと暗号”」を参照してください。

プライベートキー

サードパーティー認証機関 (CA) の証明書を使用している場合、以下の点に注意してください。

- プライベートキーが証明書のパブリックキーと一致している。
- プライベートキーは、PEM 形式でなければなりません。
- プライベートキーはパスワードで暗号化できません。

AWS Certificate Manager (ACM) が証明書を提供した場合、ACM はプライベートキーをリリースしません。プライベートキーは、ACM に統合された AWS のサービスによる使用のために、ACM に保存されます。

アクセス許可

SSL/TLS 証明書を使用およびインポートするアクセス許可が必要です。AWS Certificate Manager (ACM) を使用している場合は、AWS Identity and Access Management アクセス許可を使用して証明書へのアクセスを制限することをお勧めします。詳細については、『AWS Certificate Manager ユーザーガイド』の「Identity and Access Management」を参照してください。

証明書キーのサイズ

CloudFront がサポートする証明書キーのサイズは、キーと証明書の種類によって異なります。

RSA 証明書の場合:

CloudFront は、1024 ビット、2048 ビット、3072 ビット、4096 ビットの RSA キーをサポートしています。CloudFront で使用できる RSA 証明書のキーの長さは最大 4096 ビットです。

ACM が発行する RSA 証明書のキーの最大長は 2048 ビットであることに注意してください。3072 ビットまたは 4096 ビットの RSA 証明書を使用するには、証明書を外部から取得して ACM にインポートする必要があります。これにより、CloudFront で使用できるようになります。

RSA キーのサイズを確認する方法については、「SSL/TLS RSA 証明書内のパブリックキーのサイズを確認する」を参照してください。

ECDSA 証明書の場合:

CloudFront は、256 ビットキーをサポートしています。ACM の ECDSA 証明書を使用して、ビューワーと CloudFront との間で HTTPS を必須にするには、prime256v1 楕円曲線を使用します。

サポートされている証明書のタイプ

CloudFront は、信頼された認証機関によって発行されたすべてのタイプの証明書をサポートしています。

証明書の有効期限切れと更新

サードパーティー認定権限 (CA) から取得した証明書を使用する場合、証明書の有効期限切れをモニタリングし、AWS Certificate Manager (ACM) にインポートする、または AWS Identity and Access Management 証明書ストアにアップロードする証明書を有効期限前に更新する必要があります。

ACM が提供する証明書を使用している場合、証明書の更新は ACM によって管理されます。詳細については、AWS Certificate Manager ユーザーガイド の「管理された更新」を参照してください。

CloudFront ディストリビューションと証明書のドメイン名

カスタムオリジンを使用する場合、オリジンの SSL/TLS 証明書の共通名フィールドにドメイン名が含まれ、さらにサブジェクト代替名フィールドにもドメイン名がいくつか含まれることがあります。(CloudFront では証明書ドメイン名にワイルドカード文字を使用できます)。

証明書のドメイン名のうち 1 つは、オリジンドメイン名に指定したドメイン名と一致する必要があります。ドメイン名が一致しない場合、CloudFront は HTTP ステータスコード 502 (Bad Gateway) をビューワーに返します。

Important

ディストリビューションに代替ドメイン名を追加する場合、CloudFront はこの代替ドメイン名が添付した証明書の対象であることを確認します。証明書は、証明書のサブジェクト代替名 (SAN) フィールドの代替ドメイン名をカバーする必要があります。つまり、SAN フィールドは、代替ドメイン名と完全に一致するか、追加する代替ドメイン名の同じレベルにワイルドカードが含まれている必要があります。
詳細については、「代替ドメイン名を使用するための要件」を参照してください。

SSL/TLS プロトコルの最小バージョン

専用 IP アドレスを使用している場合、セキュリティポリシーを選ぶことでビューワーと CloudFront との間の接続に最小 SSL/TLS プロトコルバージョンを設定できます。

詳細については、トピック「ディストリビューション設定リファレンス」の「セキュリティポリシー (SSL/TLS の最小バージョン)」を参照してください。

サポートされる HTTP バージョン

1 つの証明書を複数の CloudFront ディストリビューションに関連付ける場合、証明書に関連付けられたすべてのディストリビューションで、サポートされる HTTP バージョン に対して同じオプションを使用する必要があります。このオプションは、CloudFront ディストリビューションを作成または更新するときに指定します。

CloudFront で SSL/TLS 証明書を使用する場合のクォータ (ビューワーと CloudFront との間の HTTPS のみ)

CloudFront で SSL/TLS 証明書を使用する場合は、以下のクォータに注意してください。これらのクォータは、AWS Certificate Manager (ACM) を使ってプロビジョニングした SSL/TLS 証明書や、ビューワーと CloudFront との間の HTTPS 通信のために ACM にインポートしたか IAM 証明書ストアにアップロードした SSL/TLS 証明書にのみ適用されます。

詳細については、「SSL/TLS 証明書のクォータを引き上げる」を参照してください。

CloudFront ディストリビューションあたりの証明書の最大数

各 CloudFront ディストリビューションに関連付けることができる SSL/TLS 証明書は最大 1 個です。

ACM へのインポートまたは IAM 証明書ストアへのアップロードが可能な証明書の最大数

サードパーティー認証機関から SSL/TLS 証明書を取得した場合、次のいずれかの場所に証明書を保存する必要があります。

- AWS Certificate Manager - ACM 証明書数に対する現在のクォータについては、AWS Certificate Manager ユーザーガイドの「クォータ」を参照してください。一覧表示されているクォータは、ACM を使用してプロビジョニングした証明書や ACM にインポートした証明書を含む合計です。
- IAM 証明書ストア – AWS アカウントの IAM 証明書ストアにアップロードできる証明書の数の現在のクォータ (以前は制限と呼ばれていました) については、IAM ユーザーガイドの「IAM お

よび STS の制限」を参照してください。AWS Management Console では、より高いクォータをリクエストできます。

AWS アカウントごとの証明書の最大数 (専用 IP アドレスのみ)

専用 IP アドレスを使用して HTTPS リクエストを供給する場合は、以下の点に注意してください。

- デフォルトの場合、CloudFront では 1 つの AWS アカウントで 2 つの証明書を使用する許可が与えられます。1 つは日常的に使用する証明書で、もう 1 つは複数のディストリビューションで証明書を更新する必要がある場合の証明書です。
- AWSアカウントに 2 つ以上のカスタム SSL/TLS 証明書が必要な場合は、サポートセンターにアクセスしてケースを作成します。使用するためのアクセス許可が必要な証明書の数と状況をリクエストに記載してください。できる限り早くアカウントを更新します。

複数の異なる AWS アカウントを使用して作成した CloudFront ディストリビューションに同じ証明書を使用する

サードパーティーの認証機関を使用しており、異なる AWS アカウントを使用して作成した複数の CloudFront ディストリビューションで同じ証明書を使用する場合は、AWS アカウントごとに証明書を ACM にインポートするか、IAM 証明書ストアにアップロードする必要があります。

ACM から提供される証明書を使用している場合、別の AWS アカウントで作成された証明書が CloudFront で使用されるように設定することはできません。

CloudFront と AWS の他のサービスに同じ証明書を使用する

信頼された認証機関 (Comodo、DigiCert、Symantec など) から証明書を購入した場合、CloudFront や他の AWS のサービスで同じ証明書を使用できます。ACM に証明書をインポートする場合、一度インポートするだけで複数の AWS サービスで使用できます。

ACM が提供した証明書を使用している場合、証明書は ACM に格納されています。

複数の CloudFront ディストリビューションに同じ証明書を使用する

HTTPS リクエストに対応するために使用している一部またはすべての CloudFront ディストリビューションで同じ証明書を使用できます。次の点に注意してください。

- 専用 IP アドレスを使用したリクエストの処理と SNI を使用したリクエストの処理の両方に同じ証明書を使用できます。
- 各ディストリビューションに 1 個のみ証明書を関連付けることができます。
- 各ディストリビューションには、証明書の共通名フィールドまたはサブジェクト代替名フィールドにも表示される 1 つ以上の代替ドメイン名を含める必要があります。

- 専用 IP アドレスを使用して HTTPS リクエストを処理し、同じ AWS アカウントを使用してすべてのディストリビューションを作成した場合、すべてのディストリビューションで同じ証明書を使用することによりコストを大幅に削減できます。CloudFront での課金は、ディストリビューションごとではなく、証明書ごとに行われます。

  たとえば、同じ AWS アカウントを使用して 3 つのディストリビューションを作成し、3 つのディストリビューションすべてに対して同じ証明書を使用するとします。専用 IP アドレスの使用に対する 1 つの料金のみが発生します。

  ただし、専用 IP アドレスを使用して HTTPS リクエストを処理していて、同じ証明書を使用して異なる AWS アカウントに CloudFront ディストリビューションを作成している場合は、専用 IP アドレスを使用する料金が各アカウントに発生します。たとえば、3 つの異なる AWS アカウントを使用して 3 つのディストリビューションを作成し、3 つのすべてのディストリビューションに同じ証明書を使用する場合、専用 IP アドレスを使用するための料金全額が各アカウントに課金されます。

代替ドメイン名と HTTPS を設定する

ファイルの URL で代替ドメイン名を使用し、ビューワーと CloudFront との間で HTTPS を使用するには、該当する手順を実行します。

トピック

SSL/TLS 証明書を取得する

SSL/TLS 証明書を取得します (まだ取得していない場合)。詳細については、該当するドキュメントを参照してください。

- AWS Certificate Manager (ACM) が提供する証明書を使用するには、AWS Certificate Manager ユーザーガイドを参照してください。その後、CloudFront ディストリビューションを更新するに進みます。

> **Note**
>
> ACM を使用して AWS が管理するリソースに SSL/TLS 証明書のプロビジョニング、管理、デプロイを行うことをお勧めします。米国東部 (バージニア北部) リージョンで ACM 証明書をリクエストする必要があります。

- サードパーティー認証機関 (CA) から証明書を取得するには、認証機関から提供されたドキュメントを参照してください。証明書を取得した後、次の手順に進んでください。

SSL/TLS 証明書をインポートする

サードパーティー認証機関から証明書を取得した場合、証明書を ACM にインポートするか、IAM 証明書ストアにアップロードします。

ACM (推奨)

ACM では、ACM コンソールから、またはプログラムによってサードパーティー証明書をインポートできます。ACM への証明書のインポートについては、AWS Certificate Manager ユーザーガイドの「AWS Certificate Manager への証明書のインポート」を参照してください。米国東部 (バージニア北部) リージョンで証明書をインポートする必要があります。

IAM 証明書ストア

(非推奨) 次の AWS CLI コマンドを使用してサードパーティー証明書を IAM 証明書ストアにアップロードします。

```
aws iam upload-server-certificate \
        --server-certificate-name CertificateName \
        --certificate-body file://public_key_certificate_file \
        --private-key file://privatekey.pem \
        --certificate-chain file://certificate_chain_file \
        --path /cloudfront/path/
```

次の点に注意してください。

- AWS アカウント - 証明書は、CloudFront ディストリビューションを作成するために使用した AWS アカウントと同じアカウントを使用して IAM 証明書ストアにアップロードする必要があります。

- --path パラメータ – 証明書を IAM にアップロードする場合、--path パラメータ (証明書のパス) の値が /cloudfront/ で始まる必要があります (/cloudfront/production/、/cloudfront/test/ など)。パスは / で終わる必要があります。
- 既存の証明書 – --server-certificate-name および --path パラメータには、既存の証明書に関連付けられている値とは異なる値を指定する必要があります。
- CloudFront コンソールの使用 – --server-certificate-name で AWS CLI パラメータに指定する値 (myServerCertificate など) は、CloudFront コンソールの [SSL Certificate] (SSL 証明書) リストに表示されます。
- CloudFront API の使用 – AWS CLI から返される英数字の文字列 (AS1A2M3P4L5E67SIIXR3J など) を書き留めておきます。これは、IAMCertificateId 工レメントに指定する値です。CLI から返される IAM ARN を書き留めておく必要はありません。

AWS CLI の詳細については、AWS Command Line Interface ユーザーガイドおよび AWS CLI コマンドリファレンスを参照してください。

CloudFront ディストリビューションを更新する

ディストリビューションの設定を更新するには、以下の手順を実行します。

代替ドメイン名用に CloudFront ディストリビューションを構成するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 更新するディストリビューションの ID を選択します。
3. [General] タブで、[Edit] を選択します。
4. 以下の値を更新します。

   代替ドメイン名 (CNAME)

   [項目を追加] を選択して、該当する代替ドメイン名を追加します。ドメイン名をコンマで区切るか、新しい行にドメイン名を 1 つずつ入力します。

   カスタム SSL 証明書

   ドロップダウンリストから証明書を選択します。

   100 個までの証明書がここに一覧表示されます。100 個を超える証明書があり、追加する証明書が表示されない場合には、フィールドに証明書の ARN を入力して選択できます。

IAM 証明書ストアに証明書をアップロードしたが、一覧に表示されず、フィールドに入力しても選択できない場合には、「SSL/TLS 証明書をインポートする」の手順を参照して、証明書を正しくアップロードしたかを確認します。

> **Important**
>
> SSL/TLS 証明書を CloudFront ディストリビューションに関連付けたら、すべてのディストリビューションから証明書を削除してすべてのディストリビューションがデプロイ済みになるまでは、証明書を ACM または IAM 証明書ストアから削除しないでください。

5. [Save changes] (変更の保存) をクリックします。
6. ビューワーと CloudFront の間で HTTPS が必須になるよう CloudFront を設定する

   a. [Behaviors (動作)] タブで、更新するキャッシュ動作を選択して、[Edit (編集)] を選択します。

   b. [Viewer Protocol Policy] として次のいずれかの値を指定します。

      Redirect HTTP to HTTPS

      ビューワーは両方のプロトコルを使用できますが、HTTP リクエストは自動的に HTTPS リクエストにリダイレクトされます。CloudFront は新しい HTTPS URL と一緒に HTTP ステータスコード `301 (Moved Permanently)` を返します。ビューワーはこの HTTPS URL を使用して CloudFront にリクエストを再送信します。

      > **Important**
      >
      > CloudFront は、HTTP から HTTPS に `DELETE`、`OPTIONS`、`PATCH`、`POST`、または `PUT` リクエストをリダイレクトしません。HTTPS にリダイレクトするようにキャッシュ動作を設定した場合、CloudFront は HTTP ステータスコード `DELETE` を使用して、そのキャッシュ動作の HTTP リクエスト (`OPTIONS`、`PATCH`、`POST`、`PUT`、または `403 (Forbidden)`) に応答します。

      ビューワーが作成した HTTP リクエストが HTTPS リクエストにリダイレクトされた場合、CloudFront では両方のリクエストに対する課金が発生します。HTTP リクエストの場合は、リクエストの料金と、CloudFront がビューワーに返すヘッダーの料金のみが

課金されます。HTTPS リクエストの場合、リクエストの料金と、ファイルが返すヘッダーとオブジェクトの料金が課金されます。

HTTPS Only

ビューワーは、HTTPS を使用している場合にのみ、コンテンツにアクセスできます。ビューワーから HTTPS リクエストではなく HTTP リクエストが送信された場合、CloudFront は HTTP ステータスコード 403 (Forbidden) を返し、ファイルは返しません。

c. [Yes, Edit (はい、編集します)] を選択します。

d. ビューワーと CloudFront との間で HTTPS を必須にする追加のキャッシュ動作ごとに、ステップ a から c を繰り返します。

7. 本番環境で更新された情報を使用する前に、次を確認してください。

- ビューワーに HTTPS の使用が必要とされるリクエストにのみ、各キャッシュ動作のパスパターンが適用されている。
- CloudFront が評価する順番にキャッシュ動作がリストされている。詳細については、「パスパターン」を参照してください。
- キャッシュ動作は、リクエストを正しいオリジンにルーティングします。

SSL/TLS RSA 証明書内のパブリックキーのサイズを確認する

CloudFront 代替ドメイン名と HTTPS を使用する場合、SSL/TLS RSA 証明書内のパブリックキーの最大サイズは 4096 ビットです。(これはキーサイズであり、パブリックキー内の文字数ではありません)。証明書に AWS Certificate Manager を使用する場合、ACM ではこれより大きい RSA キーがサポートされますが、CloudFront ではそのキーを使用できません。

RSA パブリック (公開) キーのサイズを確認するには、次の OpenSSL コマンドを実行できます。

```
openssl x509 -in path and filename of SSL/TLS certificate -text -noout
```

各パラメータの意味は次のとおりです。

- `-in` は、SSL/TLS RSA 証明書のパスとファイル名を指定します。
- `-text` によって、OpenSSL において、RSA パブリック (公開) キーの長さがビット単位で表示されます。
- `-noout`: OpenSSL において、パブリックキーが非表示になります。

出力例:

```
Public-Key: (2048 bit)
```

SSL/TLS 証明書のクォータを引き上げる

AWS Certificate Manager (ACM) にインポートしたり、AWS Identity and Access Management (IAM) にアップロードしたりできる SSL/TLS 証明書の数にはクォータがあります。専用 IP アドレスを使用して HTTPS リクエストを処理するように CloudFront を設定するときに AWS アカウントで使用できる SSL/TLS 証明書の数にもクォータがあります。ただし、これらのクォータ数を増やすようにリクエストできます。

トピック

- ACM にインポートする証明書のクォータを引き上げる
- IAM にアップロードする証明書のクォータを引き上げる
- 専用 IP アドレスで使用する証明書のクォータを引き上げる

ACM にインポートする証明書のクォータを引き上げる

ACM にインポートできる証明書の数のクォータについては、AWS Certificate Manager ユーザーガイドの「クォータ」を参照してください。

クォータの引き上げをリクエストするには、サポートセンターコンソールでケースを作成してください。次の値を指定します。

- [Service Limit Increase] のデフォルト値を受け入れます。
- [Limit type] で、[Certificate Manager] を選択します。
- [Region で、AWS 証明書をインポートするリージョンを指定します。
- [Limit で、 [(ACM) Number of ACM Certificates] を選択します。

次にフォームの残りに入力して送信します。

IAM にアップロードする証明書のクォータを引き上げる

IAM にアップロードできる証明書の数のクォータについては、IAM ユーザーガイドの「IAM と STS の制限」を参照してください。

クォータの引き上げをリクエストするには、サポートセンターコンソールでケースを作成してください。次の値を指定します。

- [Service Limit Increase] のデフォルト値を受け入れます。
- [Limit type] で、[Certificate Manager] を選択します。
- [Region で、AWS 証明書をインポートするリージョンを指定します。
- [Limit] で、[Server Certificate Limit (IAM) (サーバー証明書の制限 (IAM))] を選択します。

次にフォームの残りに入力して送信します。

専用 IP アドレスで使用する証明書のクォータを引き上げる

専用 IP アドレスを使用して HTTPS リクエストを処理するときに各 AWS アカウントで使用できる SSL 証明書の数のクォータについては、「SSL 証明書のクォータ」を参照してください。

クォータの引き上げをリクエストするには、サポートセンターコンソールでケースを作成してください。次の値を指定します。

- [Service Limit Increase] のデフォルト値を受け入れます。
- [Limit Type] で、[CloudFront distributions (CloudFront ディストリビューション)] を選択します。
- [Limit Type] で、 [Dedicated IP SSL Certificate Limit per Account (アカウントごとの専用 IP SSL 証明書制限)] を選択します。

次にフォームの残りに入力して送信します。

SSL/TLS 証明書をローテーションする

AWS Certificate Manager (ACM) が提供する証明書を使用する場合、SSL/TLS 証明書を更新する必要はありません。証明書の更新は、ACM によって行われます。詳細については、AWS Certificate Manager ユーザーガイドの「マネージド更新」を参照してください。

> **Note**
>
> ACM では、サードパーティー認証機関から入手して ACM にインポートした証明書の更新が管理されません。

サードパーティー認証機関を利用しており、証明書を ACM にインポートした場合 (推奨) または IAM 証明書ストアにアップロードした場合、証明書の交換が必要になることがあります。たとえば、証明書の有効期限が近づいている場合、証明書を交換する必要があります。

> **Important**
>
> 専用 IP アドレスを使用して HTTPS リクエストを処理するよう CloudFront を設定した場合、証明書をローテーションしている間、追加の証明書に対して按分された追加料金がかかる可能性があります。ディストリビューションの更新を迅速に行って、追加料金を最低限にすることをお勧めします。

SSL/TLS 証明書をローテーションする

証明書をローテーションするには、以下の手順を実行します。ビューワーは、プロセスが完了した後だけでなく、証明書を更新している間もコンテンツにアクセスし続けることができます。

SSL/TLS 証明書をローテーションするには

1. SSL/TLS 証明書のクォータを引き上げる、追加の SSL 証明書を使用するためにアクセス権限が必要かどうかを調べます。その場合、アクセス権限をリクエストし、ステップ 2 を続ける前にアクセス権限が付与されるまで待ってください。
2. 新しい証明書を ACM にインポートするか、IAM にアップロードします。詳細については、Amazon CloudFront 開発者ガイドの「SSL/TLS 証明書をインポートする」を参照してください。
3. ディストリビューションを一度に 1 つずつ、新しい証明書を使用できるように更新します。詳細については、Amazon CloudFront 開発者ガイドの「CloudFront ディストリビューションの一覧表示、詳細表示、および更新を行う」をご覧ください。
4. (オプション) すべての CloudFront ディストリビューションの更新が完了したら、古い証明書を ACM または IAM から削除することもできます。

> **Important**
>
> すべてのディストリビューションから SSL/TLS 証明書を削除し、更新されたディストリビューションのステータスが [Deployed] に変わるまで、SSL/TLS 証明書を削除しないでください。

カスタム SSL/TLS 証明書からデフォルト CloudFront 証明書に戻す

ビューワーと CloudFront との間で HTTPS を使用するように CloudFront を設定し、カスタム SSL/TLS 証明書を使用するように CloudFront を設定した場合、デフォルトの CloudFront SSL/TLS 証明書を使用するように設定を変更できます。プロセスは、コンテンツの配信にディストリビューションを使用しているかどうかによって異なります。

- ディストリビューションを使用してコンテンツを配信したことがない場合は、単に構成を変更できます。詳細については、「ディストリビューションを更新する」を参照してください。
- ディストリビューションを使用してコンテンツを配信していた場合は、新しい CloudFront ディストリビューションを作成し、コンテンツが使用できない時間を減らすかゼロにするために、ファイルの URL を変更する必要があります。そのためには、以下の手順を実行します。

デフォルト CloudFront 証明書に戻す

次の手順では、カスタム SSL/TLS 証明書からデフォルト CloudFront 証明書に戻す方法を示します。

デフォルトの CloudFront 証明書に戻すには

1. 適切な構成で、新しい CloudFront ディストリビューションを作成します。[SSL 証明書] には、[デフォルトの CloudFront 証明書 (*.cloudfront.net)] を選択します。

   詳細については、「ディストリビューションを作成する」を参照してください。
2. CloudFront を使用して配信しているファイルの場合は、アプリケーションの URL を更新して、CloudFront によって新しいディストリビューションに割り当てられたドメイン名を使用します。たとえば、`https://www.example.com/images/logo.png` を `https://d111111abcdef8.cloudfront.net/images/logo.png` に変更します。
3. 独自 SSL/TLS 証明書に関連付けられているディストリビューションを削除するか、ディストリビューションを更新して [SSL 証明書] の値を [デフォルトの CloudFront 証明書 (*.cloudfront.net)] に変更します。詳細については、「ディストリビューションを更新する」を参照してください。

   > **Important**
   >
   > このステップが完了するまで、AWS は独自 SSL/TLS 証明書を使用する料金を課金し続けます。

4. (オプション) カスタム SSL/TLS 証明書を削除します。

   a. AWS CLI コマンド `list-server-certificates` を実行して、削除する証明書の証明書 ID を取得します。詳細については、AWS CLI コマンドリファレンスの「list-server-certificates」を参照してください。

   b. AWS CLI コマンド `delete-server-certificate` を実行して、証明書を削除します。詳細については、「AWS CLI コマンドリファレンス」の「delete-server-certificate」を参照してください。

専用 IP アドレスを使用するカスタム SSL/TLS 証明書を SNI に切り替える

専用 IP アドレスで独自 SSL/TLS 証明書を使用するように CloudFront を設定した場合、SNI でカスタム SSL/TLS 証明書を使用するように切り替えて、専用 IP アドレスに関連付けられた請求が発生しないように変更できます。以下の手順でその方法を説明します。

> **Important**
>
> CloudFront 設定をこのように更新しても、SNI をサポートするビューワーには影響しません。ビューワーでは、変更前も変更後もコンテンツにアクセスできます。また、変更が CloudFront エッジロケーションに反映されている間もアクセスできます。SNI をサポートしていないビューワーは、変更後、コンテンツにアクセスできなくなります。詳細については、「CloudFront で HTTPS リクエストを処理する方法を選択する」を参照してください。

カスタム証明書から SNI に切り替える

次の手順では、専用 IP アドレスを使用するカスタム SSL/TLS 証明書を SNI に切り替える方法を示します。

独自 SSL/TLS 証明書を専用 IP アドレスから SNI に切り替えるには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 表示または更新するディストリビューションの ID を選択します。
3. [Distribution Settings] を選択します。
4. [General] タブで、[Edit] を選択します。

5. [Custom SSL Client Support (独自 SSL クライアントサポート)] の設定を [Only Clients that Support Server Name Indication (SNI) (Server Name Indication (SNI) をサポートするクライアントのみ)] に変更します。
6. [Yes, Edit (はい、編集します)] を選択します。

署名付き URL と署名付き Cookie を使用したプライベートコンテンツを提供する

インターネット経由でコンテンツを配信する多くの企業が、選ばれたユーザー (料金を支払っているユーザーなど) のドキュメント、ビジネスデータ、メディアストリーム、またはコンテンツに対して、アクセスを制限する必要があると考えています。CloudFront を使用してこのプライベートコンテンツを安全に供給するには、以下の方法を使用できます。

- 特別な CloudFront 署名付き URL または署名付き Cookie を使用してプライベートコンテンツにアクセスするようユーザーに要求します。
- オリジンサーバー (Amazon S3 やプライベート HTTPサーバーなど) 上のコンテンツに直接アクセスするURLではなく、CloudFront の URLを使用してコンテンツにアクセスするようユーザーに要求します。CloudFront URL を要求する必要はありませんが、ユーザーが署名付き URL や署名付き Cookie で指定された制限をバイパスすることを防ぐため、この方法をお勧めします。

詳細については、「ファイルへのアクセスを制限する」を参照してください。

プライベートコンテンツを提供する方法

プライベートコンテンツを供給するように CloudFront を構成するには、以下のタスクを実行します。

1. (オプション。ただし、推奨) ユーザーが CloudFront 経由でのみコンテンツにアクセスするようにします。使用する方法は、Amazon S3 を使用するかカスタムオリジンを使用するかによって異なります。
   - Amazon S3 – 「the section called “Amazon Simple Storage Service オリジンへのアクセスを制限する”」を参照してください。
   - カスタムオリジン – 「カスタムオリジンのファイルへのアクセスを制限する」を参照してください。

カスタムオリジンには、Amazon EC2、ウェブサイトエンドポイントとして設定されている Amazon S3 バケット、Elastic Load Balancing、独自の HTTP ウェブサーバーなどがあります。

2. 署名付き URL または署名付き Cookie の作成に使用する信頼されたキーグループまたは信頼された署名者を指定します。信頼されたキーグループを使用することをお勧めします。詳細については、「署名付き URL と署名付き Cookie を作成できる署名者を指定する」を参照してください。
3. 署名付き URL または署名付き Cookie を設定する `Set-Cookie` ヘッダーを使用した、承認されたユーザーからのリクエストに対応するアプリケーションを記述します。以下のいずれかのトピックの手順に従ってください。

   - 署名付き URL を使用する
   - 署名付き Cookie を使用する

   使用する方法が明確でない場合は、「署名付き URL を使用するか、署名付き Cookie を使用するかを決定する」を参照してください。

トピック

ファイルへのアクセスを制限する

プライベートコンテンツへのユーザーアクセスは 2 つの方法を使用して制御可能です。

- CloudFront キャッシュ内のファイルへのアクセスを制限します。
- 次のいずれかを実行して、オリジン内のファイルへのアクセスを制限します。
  - Amazon S3 バケットのオリジンアクセスコントロール (OAC) のセットアップ

- プライベート HTTP サーバー (カスタムオリジン) のカスタムヘッダーの設定

CloudFront キャッシュ内のファイルへのアクセスを制限する

ユーザーがファイルにアクセスする際に、署名付き URL または署名付き Cookie の使用が求められるように CloudFront を設定することができます。次に、署名付き URL を作成して認証されたユーザーに配信するか、認証されたユーザーの署名付き Cookie を設定する `Set-Cookie` ヘッダーを送信するアプリケーションを開発します (少数のファイルへの長期的なアクセスを数人のユーザーに付与するために、署名付き URL を手動で作成することもできます)。

ファイルへのアクセスを制御するための署名付き URL または署名付き Cookie を作成するときに、次の制限を指定できます。

- 最終日時。この日時以降、URL が有効ではなくなります。
- (オプション) URL が有効になる日時。
- (オプション) コンテンツへのアクセスに使用可能なコンピュータの IP アドレスまたはアドレス範囲。

署名付き URL または署名付き Cookie では、パブリックとプライベートのキーペアのプライベートキーを使用して、一部がハッシュ化され、署名が行われます。ユーザーが署名付き URL や署名付き Cookie を使用してファイルにアクセスすると、CloudFront は URL や Cookie の署名部分と無署名部分を比較します。これらが一致しない場合、CloudFront はファイルを供給しません。

URL または Cookie の署名には、RSA-SHA1 を使用する必要があります。CloudFront では他のアルゴリズムを使用できません。

Amazon S3 バケット内のファイルへのアクセスを制限する

オプションで、Amazon S3 バケット内のコンテンツを保護することで、ユーザーが指定された CloudFront ディストリビューションを介してアクセスできても、Amazon S3 URL を使用して直接アクセスすることはできないように設定できます。これにより、アクセスを制限するコンテンツについては、CloudFront のバイパスと Amazon S3 URL の使用による取得を防止できます。署名付き URL を使用するためにこの手順を実行する必要はありませんが、推奨します。

CloudFront URL を使用してコンテンツにアクセスすることをユーザーに要求するには、次のタスクを実行します。

- CloudFront に、S3 バケット内のファイルを読み取るオリジンアクセスコントロールアクセス許可を付与します。
- オリジンアクセスコントロールを作成し、CloudFront ディストリビューションに関連付けます。
- Amazon S3 URL を使用してファイルを読み取るためのアクセス許可を、他のすべてのユーザーから削除します。

詳細については、「the section called “Amazon Simple Storage Service オリジンへのアクセスを制限する”」を参照してください。

カスタムオリジンのファイルへのアクセスを制限する

カスタムオリジンを使用する場合は、カスタムヘッダーをオプションで設定して、アクセスを制限できます。CloudFront がカスタムオリジンからファイルを取得するには、標準の HTTP (または HTTPS) リクエストを使用して CloudFront からファイルにアクセスできる必要があります。しかし、カスタムヘッダーを使用することで、コンテンツへのアクセスをさらに制限して、ユーザーが直接アクセスするのではなく CloudFront を通してのみアクセスできるようにすることができます。署名付き URL を使用するためにこの手順を実行する必要はありませんが、推奨します。

ユーザーに CloudFront を経由してコンテンツにアクセスするよう要求するには、CloudFront ディストリビューションで次の設定を変更します。

オリジンのカスタムヘッダー

カスタムヘッダーがオリジンに転送されるように CloudFront を設定します。「オリジンリクエストにカスタムヘッダーを追加するように CloudFront を設定する」を参照してください。

ビューワープロトコルポリシー

ビューワーから CloudFront へのアクセス時に HTTPS の使用が求められるようにディストリビューションを設定します。「ビューワープロトコルポリシー」を参照してください。

オリジンプロトコルポリシー

CloudFront がリクエストをオリジンに転送する際にビューワーと同じプロトコルの使用が求められるように、ディストリビューションを設定します。「プロトコル (カスタムオリジンのみ)」を参照してください。

これらの変更を行った後、CloudFront で送信するように設定したカスタムヘッダーを含むリクエストのみを受け入れるように、カスタムオリジンでアプリケーションを更新します。

ビューワープロトコルポリシーとオリジンプロトコルポリシーの組み合わせにより、カスタムヘッダーが転送中に暗号化されます。ただし、定期的に以下を実行して、CloudFront がオリジンに転送するカスタムヘッダーをローテーションすることをお勧めします。

1. CloudFront ディストリビューションを更新して、カスタムオリジンへの新しいヘッダーの転送を開始します。
2. アプリケーションを更新して、リクエストが CloudFront からのものであることの確認として新しいヘッダーを受け入れます。
3. 置き換えるヘッダーが今後リクエストに含まれないようにする場合は、アプリケーションを更新して、リクエストが CloudFront からのものであることの確認として古いヘッダーを受け入れないようにします。

署名付き URL と署名付き Cookie を作成できる署名者を指定する

トピック

- 信頼されたキーグループ (推奨) と AWS アカウントのいずれかを選択する
- 署名者のキーペアを作成する
- プライベートキーの形式を変更する (.NET および Java のみ)
- ディストリビューションに署名者を追加する
- キーペアの更新

署名付き URL または署名付き Cookie を作成するには、署名者が必要です。署名者は、CloudFront で作成した信頼されたキーグループ、または CloudFront のキーペアを含む AWS アカウントのいずれかです。署名付き URL と署名付き Cookie が有効な信頼されたキーグループを使用することをお勧めします。詳細については、「信頼されたキーグループ (推奨) と AWS アカウントのいずれかを選択する」を参照してください。

署名者には 2 つの目的があります。

- 署名者をディストリビューションに追加するとすぐに、ビューワーからファイルへのアクセスには、署名付き URL または署名付き Cookie の使用が CloudFront によって求められるようになります。
- 署名付き URL または署名付き Cookie を作成するときは、署名者のキーペアのプライベートキーを使用して URL または Cookie に署名します。制限されたファイルがリクエストされると、CloudFront は URL または Cookie の署名を署名のない URL または Cookie と比較し、改ざん

されていないことを確認します。CloudFront は、URL または Cookie が有効であることも確認します。たとえば、有効期限切れ日時が経過していないことを確認します。

署名者を指定するときは、署名者をキャッシュ動作に追加することにより、署名付き URL または署名付き Cookie を必要とするファイルも間接的に指定します。ディストリビューションのキャッシュ動作が 1 つしかない場合、ビューワーはディストリビューション内のファイルへのアクセスに、署名付き URL または署名付き Cookie の使用を求められます。複数のキャッシュ動作を作成して、署名者を一部のキャッシュ動作に追加し、それ以外のキャッシュ動作に追加しなかった場合、ビューワーは一部のファイルへのアクセスに、署名付き URL または署名付き Cookie の使用を求められ、その他のファイルへのアクセスには求められません。

署名付き URL または署名付き Cookie の作成を許可する署名者 (プライベートキー) を指定し、署名者を CloudFront ディストリビューションに追加するには、以下のタスクを実行します。

1. 署名者として、信頼されたキーグループを使用するか、AWS アカウントを使用するかを決定します。信頼されたキーグループを使用することをお勧めします。詳細については、「信頼されたキーグループ (推奨) と AWS アカウントのいずれかを選択する」を参照してください。
2. ステップ 1 で選択した署名者に対して、パブリックとプライベートのキーペアを作成します。詳細については、「署名者のキーペアを作成する」を参照してください。
3. 署名付き URL または署名付き Cookie の作成に .NET または Java を使用する場合は、プライベートキーの形式を変更します。詳細については、「プライベートキーの形式を変更する (.NET および Java のみ)」を参照してください。
4. 署名付き URL または署名付き Cookie を作成するディストリビューションで、署名者を指定します。詳細については、「ディストリビューションに署名者を追加する」を参照してください。

信頼されたキーグループ (推奨) と AWS アカウントのいずれかを選択する

署名付き URL または署名付き Cookie を使用するには、署名者が必要です。署名者は、CloudFront で作成した信頼されたキーグループ、または CloudFront のキーペアを含む AWS アカウントのいずれかです。以下の理由から、信頼されたキーグループを使用することをお勧めします。

- CloudFront キーグループでは、CloudFront の署名付き URL と署名付き Cookie のパブリックキーを管理するために AWS アカウントのルートユーザーを使用する必要がありません。AWS のベストプラクティスでは、特に必要なければルートユーザーを使用しないことが推奨されています。
- CloudFront キーグループを使用すると、CloudFront API を使用して、パブリックキー、キーグループ、および信頼された署名者を管理できます。API を使用して、キーの作成とキーの更新を自

動化できます。AWS ルートユーザーを使用する場合は、AWS Management Console を使用して CloudFront キーペアを管理する必要があるため、プロセスを自動化することはできません。

- CloudFront API を使用してキーグループを管理できるため、AWS Identity and Access Management (IAM) の許可ポリシーを使用して、異なるユーザーに許可される操作を制限することもできます。例えば、ユーザーにパブリックキーのアップロードを許可し、削除を禁止することができます。または、ユーザーにパブリックキーの削除を許可するが、許可するのは、多要素認証の使用、特定のネットワークからのリクエストの送信、特定の日時範囲内でのリクエストの送信など、特定の条件が満たされた場合に限ることもできます。
- CloudFront キーグループを使用すると、より多くのパブリックキーを CloudFront ディストリビューションに関連付けることができるため、パブリックキーの使用方法と管理方法をより柔軟に設定できます。デフォルトでは、最大 4 つのキーグループを 1 つのディストリビューションに関連付けることができ、キーグループには最大 5 つのパブリックキーを含めることができます。

  AWS アカウントのルートユーザーを使用して CloudFront のキーペアを管理する場合は、AWS アカウントあたり最大 2 つのアクティブな CloudFront キーペアしか使用できません。

署名者のキーペアを作成する

CloudFront 署名付き URL または署名付き Cookie の作成に使用する各署名者には、パブリックとプライベートのキーペアが必要です。署名者はプライベートキーを使用して URL または Cookie に署名し、CloudFront はパブリックキーを使用して署名を検証します。

キーペアを作成する方法は、信頼されたキーグループを署名者として使用するか (推奨)、CloudFront のキーペアを使用するかによって異なります。詳細については、次のセクションを参照してください。作成するキーペアは、以下の要件を満たしている必要があります。

- SSH-2RSA キーペアである必要があります。
- base64 エンコードされた PEM 形式である必要があります。
- 2048 ビットのキーペアである必要があります。

アプリケーションを保護するために、キーペアを定期的に更新することをお勧めします。詳細については、「キーペアの更新」を参照してください。

信頼されたキーグループのキーペアを作成する (推奨)

信頼されたキーグループのキーペアを作成するには、以下の手順を実行します。

1. パブリックとプライベートのキーペアを作成します。

2. パブリックキーを CloudFront にアップロードします。
3. パブリックキーを CloudFront キーグループに追加します。

詳細については、次の手順を参照してください。

キーペアを作成するには

> **Note**
>
> 以下の手順では、キーペアを作成する方法の一例として OpenSSL を使用します。RSA キーペアを作成する方法は他にも多数あります。

1. 以下のコマンド例では、OpenSSL を使用して 2,048 ビット長の RSA キーペアを生成し、`private_key.pem` という名前のファイルに保存します。

   ```
   openssl genrsa -out private_key.pem 2048
   ```

2. 生成されるファイルには、パブリックキーとプライベートキーの両方が含まれます。以下のコマンド例では、`private_key.pem` という名前のファイルからパブリックキーを抽出します。

   ```
   openssl rsa -pubout -in private_key.pem -out public_key.pem
   ```

   後で、以下の手順でパブリックキー (`public_key.pem` ファイル内) をアップロードします。

パブリックキーを CloudFront にアップロードするには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションメニューで、[Public keys (パブリックキー)] を選択します。
3. [パブリックキーを作成] を選択します。
4. [パブリックキーを作成] ウィンドウで、次の操作を行います。

   a. [Key name (キー名)] に、パブリックキーを識別するための名前を入力します。

b. [Key value (キー値)] に、パブリックキーを貼り付けます。前の手順のステップに従った場合、パブリックキーは `public_key.pem` という名前のファイルにあります。パブリックキーの内容をコピーして貼り付けるには、以下の手順を実行します。

- macOS または Linux コマンドラインで cat コマンドを次のように使用します。

```
cat public_key.pem
```

そのコマンドの出力をコピーして、[Key value (キー値)] フィールドに貼り付けます。

- メモ帳 (Windows の場合) やテキストエディット (macOS の場合) などのプレーンテキストエディタで、`public_key.pem` ファイルを開きます。ファイルの内容をコピーし、[Key value (キー値)] フィールドに貼り付けます。

c. (オプション) [Comment (コメント)] に、パブリックキーを説明するコメントを追加します。

完了したら、[Add (追加)] を選択します。

5. パブリックキー ID を記録します。この ID は、後で署名付き URL または署名付き Cookie を作成するときに、`Key-Pair-Id` フィールドの値として使用します。

パブリックキーをキーグループに追加するには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。
2. ナビゲーションメニューで、[Key groups (キーグループ)] を選択します。
3. [Add key group (キーグループの追加)] を選択します。
4. [Create key group (キーグループの作成)] ページで、以下の手順を実行します。

a. [Key group name (キーグループ名)] に、キーグループを識別するための名前を入力します。

b. (オプション) [Comment (コメント}] に、キーグループを説明するコメントを入力します。

c. [Public keys (パブリックキー)] で、キーグループに追加するパブリックキーを選択してから、[Add (追加)] を選択します。キーグループに追加するパブリックキーごとに、このステップを繰り返します。

5. [Create key group (キーグループの作成)] を選択します。

6. キーグループ名を記録します。この名前は、後でキーグループを CloudFront ディストリビューションのキャッシュ動作に関連付けるときに使用します。(CloudFront API では、キーグループ ID を使用して、キーグループをキャッシュ動作に関連付けます)。

CloudFront のキーペアを作成する (非推奨、AWS アカウントのルートユーザーが必要)

> **Important**
>
> ここでの手順に従う代わりに、信頼されたキーグループのパブリックキーを作成することをお勧めします。署名付き URL および署名付き Cookie のパブリックキーを作成するための推奨される方法については、「信頼されたキーグループのキーペアを作成する (推奨)」を参照してください。

CloudFront のキーペアは、次の方法で作成できます。

- AWS Management Console でキーペアを作成し、プライベートキーをダウンロードします。後述の手順を参照してください。
- OpenSSL などのアプリケーションを使用して RSA キーペアを作成し、パブリックキーを AWS Management Console にアップロードします。RSA キーペアの作成の詳細については、「信頼されたキーグループのキーペアを作成する (推奨)」を参照してください。

AWS Management Console で CloudFront キーペアを作成するには

1. AWS アカウントのルートユーザーの認証情報を使用して、AWS Management Console にサインインします。

   > **Important**
   >
   > IAM ユーザーは CloudFront のキーペアを作成できません。キーペアを作成するには、ルートユーザーの認証情報を使用してサインインする必要があります。

2. アカウント名を選択してから、[My Security Credentials (セキュリティ認証情報)] を選択します。
3. [CloudFront のキーペア] を選択します。
4. 複数のキーペアが有効になっていないことを確認します。既に 2 つのキーペアが有効になっていると、キーペアを作成できません。

5. [Create New Key Pair (新しいキーペアの作成)] を選択します。

> **Note**
>
> 独自のキーペアを作成し、公開キーをアップロードすることもできます。CloudFront キーペアは、1024、2048、または 4096 ビットのキーをサポートします。

6. [Create Key Pair (キーペアの作成)] ダイアログボックスで、[Download Private Key File (プライベートキーファイルのダウンロード)] を選択し、ファイルをコンピュータに保存します。

> **Important**
>
> CloudFront のキーペアのプライベートキーを安全な場所に保存し、必要な管理者だけがそのプライベートキーを読み取ることができるようにファイルのアクセス許可を設定します。別のユーザーがこのプライベートキーを取得すると、そのユーザーは有効な署名付き URL および署名付き Cookie を生成し、コンテンツをダウンロードできます。プライベートキーを再取得することはできません。したがって、プライベートキーを削除したか失った場合は、新しい CloudFront キーペアを作成する必要があります。

7. キーペアのキーペア ID を記録しておきます (AWS Management Console ではアクセスキー ID と呼ばれます)。この情報は、署名付き URL または署名付き Cookie を作成するときに使用します。

プライベートキーの形式を変更する (.NET および Java のみ)

.NET または Java を使用して署名付き URL または署名付き Cookie を作成する場合、キーペアのプライベートキーをデフォルトの PEM 形式のまま使用して署名を作成することはできません。代わりに、以下の手順を実行します。

- .NET Framework – .NET Framework で使用する XML 形式にプライベートキーを変換します。いくつかのツールを利用できます。
- Java – DER 形式にプライベートキーを変換します。そのための 1 つの方法は、以下の OpenSSL コマンドを使用することです。以下のコマンドで、`private_key.pem` は PEM 形式のプライベートキーを含むファイルの名前であり、`private_key.der` はコマンドの実行後に DER 形式のプライベートキーを含むファイルの名前です。

```
openssl pkcs8 -topk8 -nocrypt -in private_key.pem -inform PEM -out private_key.der -
outform DER
```

エンコーダーが正常に機能するように、Bouncy Castle の Java 用暗号 API の JAR をプロジェクトに追加してから Bouncy Castle プロバイダーを追加します。

ディストリビューションに署名者を追加する

署名者は、ディストリビューション用の署名付き URL と署名付き Cookie を作成できる、信頼されたキーグループ (推奨) または CloudFront キーペアです。署名付き URL または署名付き Cookie を CloudFront ディストリビューションで使用するには、署名者を指定する必要があります。

署名者はキャッシュ動作に関連付けられています。これにより、同じディストリビューション内で、一部のファイルに署名付き URL または署名付き Cookie を要求し、その他のファイルには要求しないということが可能になります。ディストリビューションでは、対応するキャッシュ動作に関連付けられているファイルにのみ、署名付き URL または Cookie が必要です。

同様に、署名者は、対応するキャッシュ動作に関連付けられているファイルの URL または Cookie にのみ署名できます。例えば、1 つのキャッシュ動作に対して 1 つの署名者があり、別のキャッシュ動作に対して別の署名者がある場合、どちらの署名者も、もう一方のキャッシュ動作に関連付けられたファイルに対して署名付き URL または署名付き Cookie を作成できません。

Important

ディストリビューションに署名者を追加する前に、以下の手順を実行します。

- キャッシュ動作のパスパターンとキャッシュ動作のシーケンスを慎重に定義して、ユーザーにコンテンツへの意図しないアクセスを許可したり、すべてのユーザーを対象としたコンテンツへのアクセスを禁止したりしないようにします。

  たとえば、リクエストが、2 つのキャッシュ動作のパスパターンに一致したと仮定します。最初のキャッシュ動作は署名付き URL または署名付き Cookie を要求しませんが、2 番目のキャッシュ動作はこれらを要求します。ユーザーは署名付き URL または署名付き Cookie を使用せずにファイルにアクセスできます。これは、CloudFront が、最初の一致に関連付けられたキャッシュ動作を処理するためです。

  パスパターンの詳細については、「パスパターン」を参照してください。
- コンテンツの配信に既に使用しているディストリビューションの場合は、署名者を追加する前に、署名付き URL と署名付き Cookie の生成を開始する準備ができていることを確認

してください。署名者を追加すると、CloudFront は有効な署名付き URL または署名付き Cookie を含まないリクエストを拒否します。

CloudFront コンソールまたは CloudFront API を使用して、署名者をディストリビューションに追加できます。

Console

以下の手順では、信頼されたキーグループを署名者として追加する方法を示します。信頼された署名者として AWS アカウントを追加することもできますが、お勧めしません。

コンソールを使用してディストリビューションに署名者を追加するには

1. 信頼された署名者として使用するキーグループのキーグループ ID を記録します。詳細については、「信頼されたキーグループのキーペアを作成する (推奨)」を参照してください。
2. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。
3. 署名付き URL または署名付き Cookie でファイルを保護するディストリビューションを選択します。

   > **Note**
   >
   > 新しいディストリビューションに署名者を追加するには、ディストリビューションを作成するときにステップ 6 で説明したのと同じ設定を指定します。

4. [Behaviors] タブを選択します。
5. 署名付き URL または署名付き Cookie で保護するファイルとパスパターンが一致するキャッシュ動作を選択し、[Edit (編集)] を選択します。
6. [Edit Behavior (動作の編集)] ページで、以下の手順を実行します。

   a. [Restrict Viewer Access (Use Signed URLs or Signed Cookies) (ビューワーのアクセスを制限 (署名付き URL または署名付き Cookie の使用))] で、[はい] を選択します。

   b. [Trusted Key Groups or Trusted Signer (信頼されたキーグループまたは信頼された署名者)] で、[Trusted Key Groups (信頼されたキーグループ)] を選択します。

   c. [Trusted Key Groups (信頼されたキーグループ)] で、追加するキーグループを選択し、[Add (追加)] を選択します。複数のキーグループを追加する場合は、この手順を繰り返します。

7. [Yes, Edit (はい、編集する)] を選択して、キャッシュ動作を更新します。

API

CloudFront API を使用して、信頼されたキーグループを署名者として追加できます。署名者を既存のディストリビューションまたは新しいディストリビューションに追加できます。どちらの場合も、`TrustedKeyGroups` 要素の値を指定します。

信頼された署名者として AWS アカウントを追加することもできますが、お勧めしません。

Amazon CloudFront API リファレンスの以下のトピックを参照してください。

- 既存のディストリビューションを更新する – 「UpdateDistribution」
- 新しいディストリビューションを作成する – 「CreateDistribution」

キーペアの更新

署名付き URL と署名付き Cookie のキーペアを定期的に更新 (変更) することをお勧めします。有効期限がまだ切れていない URL または Cookie を無効にすることなく、署名付き URL または署名付き Cookie の作成に使用しているキーペアを更新するには、以下のタスクを実行します。

1. 新しいキーペアを作成し、パブリックキーをキーグループに追加します。詳細については、「信頼されたキーグループのキーペアを作成する (推奨)」を参照してください。
2. 前の手順で新しいキーグループを作成した場合は、キーグループを署名者としてディストリビューションに追加します。

> **Important**
>
> キーグループから既存のパブリックキーを削除したり、ディストリビューションからキーグループを削除したりしないでください。新規追加のみを行ってください。

3. 新しいキーペアのプライベートキーを使用して署名を作成するようにアプリケーションを更新します。新しいプライベートキーで署名された URL または Cookie が機能することを確認します。
4. 以前のプライベートキーを使用して署名付き URL または Cookie の有効期限切れ日時が経過するまで待ちます。次に、古いパブリックキーをキーグループから削除します。ステップ 2 で新しいキーグループを作成した場合は、ディストリビューションから古いキーグループを削除します。

署名付き URL を使用するか、署名付き Cookie を使用するかを決定する

CloudFront 署名付き URL と署名付き Cookie は同じ基本的な機能を提供します。これらによって、コンテンツにアクセスできるユーザーを制御できます。CloudFront を使用してプライベートコンテンツを供給する場合に、署名付き URL と署名付き Cookie のどちらを使用するかを決定するには、以下の点を考慮します。

次のような場合は、署名付き URL を使用します。

- 個別のファイル (アプリケーションのインストールダウンロード) へのアクセスを制限する場合。
- ユーザーが Cookie をサポートしていないクライアント (カスタム HTTP クライアントなど) を使用している場合。

次のような場合は、署名付き Cookie を使用します。

- 複数の制限されたファイル (HLS 形式の動画のすべてのファイルやウェブサイトの購読者の領域にあるすべてのファイルなど) へのアクセスを提供する場合。
- 現在の URL を変更したくない場合。

現在署名付き URL を使用していない場合で、署名なし URL に次のクエリ文字列パラメータ含まれる場合、署名付き URL と署名付き Cookie のいずれも使用できません。

- `Expires`
- `Policy`
- `Signature`
- `Key-Pair-Id`

CloudFront では、これらのクエリ文字列パラメータを含む URL が署名付き URL であると見なされ、署名付き Cookie の確認は行われません。

署名付き URL と署名付き Cookie の両方を使用する

署名付き URL は署名付き Cookie よりも優先されます。署名付き URL と署名付き Cookie の両方を使用して同じファイルへのアクセスが制御されている場合に、ビューワーが署名付き URL を使用してファイルをリクエストすると、CloudFront は署名付き URL のみに基づいてビューワーにファイルを返すかどうかを決定します。

署名付き URL を使用する

署名付き URL には、有効期限切れ日時など、追加の情報も含まれており、コンテンツへのアクセスをより詳細に制御できます。この追加情報は、既定ポリシーまたはカスタムポリシーに基づくポリシーステートメントに含まれます。既定ポリシーとカスタムポリシーの違いを以下の 2 つのセクションで説明します。

> **Note**
>
> 既定ポリシーを使用して一部の署名付き URL を作成し、同じディストリビューションで、カスタムポリシーを使用して一部の署名付き URL を作成することができます。

トピック

署名付き URL に既定ポリシーとカスタムポリシーのどちらを使用するかを決定する

署名付き URL を作成する場合、URL の有効期間など、署名付き URL で制限を指定する JSON 形式のポリシーステートメントを作成します。既定ポリシーまたはカスタムポリシーのいずれかを使用できます。既定ポリシーとカスタムポリシーの比較を以下に示します。

説明	既定ポリシー	カスタムポリシー
ポリシーステートメントを複数のファイル用に再利用できる。ポリシーステートメントを再利用するには、`Resource` オブジェクトでワイルドカード文字を使用する必要があります。(詳しくは、「カスタムポリシーを使用する署名付き URL のポリシーステートメントで指定する値」を参照してください)。	いいえ	はい

説明	既定ポリシー	カスタムポリシー
ユーザーがコンテンツへのアクセスを開始できる日時を指定できる。	いいえ	はい (オプション)
ユーザーがコンテンツにアクセスできなくなる日時を指定できる。	はい	はい
コンテンツにアクセスできるユーザーの IP アドレスまたは IP アドレス範囲を指定できる。	いいえ	はい (オプション)
署名付き URL にポリシーの base64 エンコードされたバージョンが含まれているため、URL が長くなる。	いいえ	はい

既定ポリシーを使用して署名付き URL を作成する方法については、「既定ポリシーを使用して署名付き URL を作成する」を参照してください。

カスタムポリシーを使用して署名付き URL を作成する方法については、「カスタムポリシーを使用して署名付き URL を作成する」を参照してください。

署名付き URL の仕組み

署名付き URL 用の CloudFront と Amazon S3 の設定方法と、ユーザーが署名付き URL を使用してファイルをリクエストしたときの CloudFront の応答方法の概要を次に示します。

1. CloudFront ディストリビューションで、1 つ以上の信頼されたキーグループを指定します。指定したグループには、CloudFront が URL 署名の検証に使用できるパブリックキーが含まれている必要があります。対応するプライベートキーを使用して URL に署名します。

   詳細については、「署名付き URL と署名付き Cookie を作成できる署名者を指定する」を参照してください。

2. アプリケーションを開発して、ユーザーがコンテンツへのアクセス許可を持つかどうかを決定し、アプリケーションでアクセスを制限するファイルまたは部分用に署名付き URL を作成するかを決定します。詳細については、以下のトピックを参照してください。

   - 既定ポリシーを使用して署名付き URL を作成する
   - カスタムポリシーを使用して署名付き URL を作成する

3. 署名付き URL を必要とするファイルをユーザーが要求します。
4. アプリケーションは、ファイルにアクセスするための資格がユーザーにあることを検証します。たとえば、ユーザーがサインインしていること、コンテンツへのアクセス料を支払っていること、他のいくつかのアクセス要件を満たしていることを検証します。
5. アプリケーションは署名付き URL を作成してエンドユーザーに返します。
6. 署名付き URL を使用すると、ユーザーはコンテンツのダウンロードやストリーミングができます。

   このステップは自動で実行されます。ユーザーは、コンテンツにアクセスする以外に、何も行う必要はありません。たとえば、ユーザーがウェブブラウザでコンテンツにアクセスすると、アプリケーションは署名付き URL をブラウザに返します。ブラウザは、直ちに、署名付き URL を使用して CloudFront エッジキャッシュ内のファイルにアクセスします。ユーザーからの介入はありません。
7. CloudFront はパブリックキーを使用して署名を検証し、URL が改ざんされていないことを確認します。署名が無効である場合、リクエストは拒否されます。

   署名が有効である場合、CloudFront は URL のポリシーステートメントを参照して (または既定ポリシーを使用している場合はポリシーステートメントを作成して)、リクエストがまだ有効であることを確認します。たとえば、URL の開始日時と終了日時を指定した場合、CloudFront は、アクセスが許可されている期間にユーザーがコンテンツへのアクセスを試みていることを確認します。

   リクエストがポリシーステートメントの要件を満たしている場合、CloudFront は標準の操作を実行します。つまり、ファイルがエッジキャッシュにすでに存在するかどうかを確認し、必要に応じてリクエストをオリジンに転送して、ファイルをユーザーに返します。

> **Note**
>
> 署名なし URL にクエリ文字列パラメータが含まれている場合は、署名する URL にそれらのパラメータを含めてください。URL への署名後にその URL にクエリ文字列を追加すると、HTTP 403 ステータスが返されます。

署名付き URL の有効期間を決定する

短期間 (おそらく数分) のみ有効な署名付き URL を使用してプライベートコンテンツを配信できます。このように短期間有効な署名付き URL は、特定の目的で、コンテンツをユーザーにオンザフラ

イで配信する場合に適しています。たとえば、映画レンタルや音楽ダウンロードをカスタマーにオンデマンドで配信する場合に適しています。署名付き URL を短期間だけ有効にする場合、開発したアプリケーションを使用して、署名付き URL を自動生成することが必要になる場合があります。ユーザーがファイルのダウンロードまたはメディアファイルの再生を開始すると、CloudFront は、URL 内の有効期限切れ時刻と現在の時刻を比較して、URL が依然として有効かどうかを判別します。

これよりも有効期間の長い (おそらく数年間の) 署名付き URL を使用して、プライベートコンテンツを配信できます。有効期間の長い署名付き URL は、プライベートコンテンツを既知のユーザーに配信する場合に役立ちます。たとえば、事業計画を投資家に配信したり、教育資料を従業員に配信したりする場合に役立ちます。これらの長期的な署名付き URL を生成するアプリケーションを開発できます。

CloudFront が署名付き URL の有効期限切れの日時を確認するタイミング

CloudFront は、HTTP リクエスト時に署名付き URL の有効期限切れ日時を確認します。有効期限切れ時刻の直前にクライアントが大きなファイルのダウンロードを開始した場合、ダウンロード中に有効期限切れ時刻が経過してもダウンロードは完了します。TCP 接続が中断し、有効期限切れ時刻が経過した後にクライアントがダウンロードを再開した場合、ダウンロードは失敗します。

クライアントが、ファイルを断片的に取得するレンジ GET を使用した場合、有効期限切れ時刻が経過した後に実行された GET リクエストは失敗します。レンジ GET の詳細については、「CloudFront がオブジェクトの部分的リクエスト (レンジ GET) を処理する方法」を参照してください。

サンプルコードおよびサードパーティーツール

署名付き URL の、ハッシュ化および署名されたパートを作成するサンプルコードについては、以下のトピックを参照してください。

- Perl を使用して URL 署名を作成する
- PHP を使用して URL 署名を作成する
- C# と .NET Framework を使用して URL 署名を作成する
- Java を使用して URL 署名を作成する

既定ポリシーを使用して署名付き URL を作成する

既定ポリシーを使用して署名付き URL を作成するには、以下の手順を実行します。

既定ポリシーを使用して署名付き URL を作成するには

1. .NET または Java を使用して署名付き URL を作成しており、キーペアのプライベートキーをデフォルトの .pem 形式から .NET または Java 対応の形式に変更していない場合は、それを変換します。詳細については、「プライベートキーの形式を変更する (.NET および Java のみ)」を参照してください。
2. 以下の値を、リストされた順に連結し、この署名付き URL の例に示されている形式を複製します。

```
https://d111111abcdef8.cloudfront.net/
image.jpg?color=red&size=medium&Expires=1357034400&Signature=nitfHRCrtziwO2HwPfWw~yYDhUF5Ew
j19DzZrvDh6hQ73lDx~-ar3UocvvRQVw6EkC~GdpGQyyOSKQim-
TxAnW7d8F5Kkai9HVx0FIu-5jcQb0UEmatEXAMPLE3ReXySpLSMj0yCd3ZAB4UcBCAqEijkytL6f3fVYNGQI6&Key-
Pair-Id=K2JCJMDEHXQW5F
```

すべての空白 (タブと改行文字を含む) を削除します。アプリケーションコード内の文字列にエスケープ文字を含めることが必要になる場合があります。すべての値の型は String です。

1. ***######## URL***

   ベース URL は、署名付き URL を使用しなかった場合にファイルへのアクセスに使用する CloudFront URL であり、独自のクエリ文字列パラメータを含みます (ある場合)。前の例で、ベース URL は https://d111111abcdef8.cloudfront.net/image.jpg です。ディストリビューション用の URL 形式の詳細については、「CloudFront でファイルの URL 形式をカスタマイズする」を参照してください。

   - 以下の CloudFront URL は、ディストリビューション内のイメージファイルの URL です (CloudFront ドメイン名を使用)。image.jpg は images ディレクトリにあります。URL 内のファイルへのパスは、HTTP サーバーまたは Amazon S3 バケットのファイルへのパスに一致する必要があります。

     https://d111111abcdef8.cloudfront.net/images/image.jpg

   - 以下の CloudFront URL には、クエリ文字列が含まれます。

     https://d111111abcdef8.cloudfront.net/images/image.jpg?size=large

   - 以下の CloudFront URL は、ディストリビューション内のイメージファイルの URL です。どちらも代替ドメイン名を使用します。2 番目にはクエリ文字列が含まれています。

     https://www.example.com/images/image.jpg

```
https://www.example.com/images/image.jpg?color=red
```

- 以下の CloudFront URL は、代替ドメイン名と HTTPS プロトコルを使用するディストリビューション内のイメージファイルの URL です。

```
https://www.example.com/images/image.jpg
```

2. **?**

   ? は、クエリ文字列パラメータがベース URL の後に続くことを示します。独自のクエリ文字列パラメータがない場合も ? を含めます。

3. ***########## (####)*** **&**

   この値はオプションです。独自のクエリ文字列パラメータ、たとえば次のクエリ文字列パラメータを追加すると仮定します。

   ```
   color=red&size=medium
   ```

   この場合は、? の後、かつ Expires パラメータの前にパラメータを追加します。特定のまれな状況では、Key-Pair-Id の後にクエリ文字列パラメータを配置する必要があります。

   > **Important**
   >
   > パラメータに Expires、Signature、または Key-Pair-Id という名前を付けることはできません。

   独自のパラメータを追加する場合は、各パラメータの後に & を付加します (最後のパラメータにも付加します)。

4. **Expires=*Unix #### (###) ######## (UTC) ###***

   URL によるファイルへのアクセスの許可を停止する日付と時刻。

   有効期限切れ日時を Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。例えば、このトピックの冒頭にある例に示すように、2013 年 1 月 1 日午前 10:00 UTC は Unix 時間形式で 1357034400 に変換されます。エポック時間を使用するには、2147483647 (2038 年 1 月 19 日 03:14:07 UTC) 以前の日付に 32 ビット整数を使用します。UTC の詳細については、RFC 3339, Date and Time on the Internet: Timestamps を参照してください。

5. **&Signature=################################**

   ハッシュ化され、署名された base64 エンコードバージョンの JSON ポリシーステートメント。詳細については、「既定ポリシーを使用する署名付き URL の署名を作成する」を参照してください。

6. **&Key-Pair-Id=########################## *CloudFront* ############### *ID***

   CloudFront パブリックキーの ID (K2JCJMDEHXQW5F など)。パブリックキー ID は、署名付き URL の検証に使用するパブリックキーを CloudFront に通知します。CloudFront は、署名内の情報をポリシーステートメント内の情報と比較して、URL が改ざんされていないことを確認します。

   このパブリックキーは、ディストリビューションの信頼された署名者であるキーグループに属している必要があります。詳細については、「署名付き URL と署名付き Cookie を作成できる署名者を指定する」を参照してください。

既定ポリシーを使用する署名付き URL の署名を作成する

既定ポリシーを使用する署名付き URL の署名を作成するには、以下の手順を実行します。

トピック

- 既定ポリシーを使用する署名付き URL のポリシーステートメントを作成する
- 既定ポリシーを使用する署名付き URL の署名を作成する

既定ポリシーを使用する署名付き URL のポリシーステートメントを作成する

既定ポリシーを使用して署名付き URL を作成する場合、Signature パラメータは、ポリシーステートメントのハッシュ化および署名されたバージョンです。カスタムポリシーを使用する署名付き URL とは異なり、既定ポリシーを使用する署名付き URL では、URL にポリシーステートメントを含めません。ポリシーステートメントを作成するには、以下の手順を実行します。

既定ポリシーを使用する署名付き URL のポリシーステートメントを作成するには

1. 以下の JSON 形式および UTF-8 文字エンコードを使用してポリシーステートメントを構築します。すべての句読点および他のリテラル値を、指定されたとおりに正確に含めます。Resource および DateLessThan パラメータの詳細については、「既定ポリシーを使用する署名付き URL のポリシーステートメントで指定する値」を参照してください。

```
{
    "Statement": [
        {
            "Resource": "base URL or stream name",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime": ending date and time in Unix time format and
 UTC
                }
            }
        }
    ]
}
```

2. ポリシーステートメントからすべての空白 (タブと改行文字を含む) を削除します。アプリケーションコード内の文字列にエスケープ文字を含めることが必要になる場合があります。

既定ポリシーを使用する署名付き URL のポリシーステートメントで指定する値

既定ポリシーのポリシーステートメントを作成する場合、以下の値を指定します。

リソース

> **Note**
>
> Resource の日付形式は 1 つだけ指定できます。

クエリ文字列 (ある場合) が含まれるベース URL。ただし、CloudFront の Expires、Signature、および Key-Pair-Id パラメータが除外されます。次に例を示します。

```
https://d111111abcdef8.cloudfront.net/images/horizon.jpg?
size=large&license=yes
```

次の点に注意してください。

- プロトコル – 値は http:// または https:// で始まっている必要があります。
- クエリ文字列パラメータ – クエリ文字列パラメータがない場合は、疑問符を省略します。

- 代替ドメイン名 – URL で代替ドメイン名 (CNAME) を指定する場合は、ウェブページまたはアプリケーション内のファイルを参照するときに代替ドメイン名を指定する必要があります。オブジェクトの Amazon S3 URL を指定しないでください。

DateLessThan

URL の有効期限切れ日時。Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。たとえば、UTC の 2013 年 1 月 1 日午前 10 時 00 分は、Unix 時間形式の 1357034400 に変換されます。

この値は、署名付き URL 内の `Expires` クエリ文字列パラメータの値と一致する必要があります。値を引用符で囲まないでください。

詳細については、「CloudFront が署名付き URL の有効期限切れの日時を確認するタイミング」を参照してください。

既定ポリシーを使用する署名付き URL のポリシーステートメントの例

署名付き URL 内の既定ポリシーの以下のポリシーステートメントの例を使用すると、ユーザーは、UTC の 2013 年 1 月 1 日午前 10 時 00 分までファイル `https://d111111abcdef8.cloudfront.net/horizon.jpg` にアクセスできます。

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/horizon.jpg?
size=large&license=yes",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime": 1357034400
                }
            }
        }
    ]
}
```

既定ポリシーを使用する署名付き URL の署名を作成する

署名付き URL の `Signature` パラメータの値を作成するには、「既定ポリシーを使用する署名付き URL のポリシーステートメントを作成する」で作成したポリシーステートメントをハッシュ化して署名します。

ポリシーステートメントのハッシュ化、署名、およびエンコードを行う方法の詳細および例については、以下の各資料を参照してください。

- Linux コマンドおよび OpenSSL を使用した Base64 エンコードおよび暗号化
- 署名付き URL の署名を作成するためのコード例

オプション 1: 既定ポリシーを使用して署名を作成するには

1. 「既定ポリシーを使用する署名付き URL のポリシーステートメントを作成するには」の手順で作成したポリシーステートメントを、SHA-1 ハッシュ関数と RSA を使用してハッシュ化し、署名します。空白を含まないバージョンのポリシーステートメントを使用します。

   ハッシュ関数に必要なプライベートキーには、対応するパブリックキーがディストリビューション内のアクティブな信頼されたキーグループにあるものを使用してください。

   > **Note**
   >
   > ポリシーステートメントをハッシュ化および署名するための方法は、プログラミング言語およびプラットフォームによって異なります。サンプルコードについては、「署名付き URL の署名を作成するためのコード例」を参照してください。

2. ハッシュ化および署名した文字列から、空白 (タブや改行文字を含む) を削除します。
3. MIME base64 エンコーディングを使用して文字列を Base64 エンコードします。詳細については、RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies の Section 6.8, Base64 Content-Transfer-Encoding を参照してください。
4. URL クエリ文字列内の無効な文字を有効な文字で置き換えます。次の表に無効な文字と有効な文字を示します。

無効な文字 (置換元)	有効な文字 (置換先)
+	- (ハイフン)
=	_ (下線)
/	~ (チルダ)

5. 結果の値を署名付き URL の `&Signature=` の後に付加し、「既定ポリシーを使用して署名付き URL を作成するには」に戻って、署名付き URL の各パートの連結を終了します。

カスタムポリシーを使用して署名付き URL を作成する

カスタムポリシーを使用して署名付き URL を作成するには、次の手順を実行します。

カスタムポリシーを使用して署名付き URL を作成するには

1. .NET または Java を使用して署名付き URL を作成しており、キーペアのプライベートキーをデフォルトの .pem 形式から .NET または Java 対応の形式に変更していない場合は、それを変換します。詳細については、「プライベートキーの形式を変更する (.NET および Java のみ)」を参照してください。
2. 以下の値を、リストされた順に連結し、この署名付き URL の例に示されている形式を複製します。

```
https://d111111abcdef8.cloudfront.net/
image.jpg?color=red&size=medium&Policy=eyANCiAgICEXAMPLEW1lbnQiOiBbeyANCiAgICAgICJSZXNvdXJj
j19DzZrvDh6hQ73lDx~-ar3UocvvRQVw6EkC~GdpGQyyOSKQim-
TxAnW7d8F5Kkai9HVx0FIu-5jcQb0UEmatEXAMPLE3ReXySpLSMj0yCd3ZAB4UcBCAqEijkytL6f3fVYNGQI6&Key-
Pair-Id=K2JCJMDEHXQW5F
```

すべての空白 (タブと改行文字を含む) を削除します。アプリケーションコード内の文字列に エスケープ文字を含めることが必要になる場合があります。すべての値の型は `String` です。

1. ***######## URL***

   ベース URL は、署名付き URL を使用しなかった場合にファイルへのアクセスに使用する CloudFront URL であり、独自のクエリ文字列パラメータを含みます (ある場合)。前の例で、ベース URL は `https://d111111abcdef8.cloudfront.net/image.jpg` です。ディストリビューション用の URL 形式の詳細については、「CloudFront でファイルの URL 形式をカスタマイズする」を参照してください。

   以下の例は、ディストリビューションで指定する値を示しています。

   - 以下の CloudFront URL は、ディストリビューション内のイメージファイルの URL です (CloudFront ドメイン名を使用)。`image.jpg` は `images` ディレクトリにあります。URL 内のファイルへのパスは、HTTP サーバーまたは Amazon S3 バケットのファイルへのパスに一致する必要があります。

     ```
     https://d111111abcdef8.cloudfront.net/images/image.jpg
     ```

   - 以下の CloudFront URL には、クエリ文字列が含まれます。

```
https://d111111abcdef8.cloudfront.net/images/image.jpg?size=large
```

- 以下の CloudFront URL は、ディストリビューション内のイメージファイルの URL です。両方の URL で代替ドメイン名が使用されており、2 番目の URL にはクエリ文字列が含まれています。

  ```
  https://www.example.com/images/image.jpg
  ```

  ```
  https://www.example.com/images/image.jpg?color=red
  ```

- 以下の CloudFront URL は、代替ドメイン名と HTTPS プロトコルを使用するディストリビューション内のイメージファイルの URL です。

  ```
  https://www.example.com/images/image.jpg
  ```

2. **?**

   ? は、クエリ文字列パラメータがベース URL の後に続くことを示します。独自のクエリ文字列パラメータがない場合も ? を含めます。

3. ***########## (####)*** **&**

   この値はオプションです。独自のクエリ文字列パラメータ、たとえば次のクエリ文字列パラメータを追加すると仮定します。

   ```
   color=red&size=medium
   ```

   この場合、? の後、かつ Policy パラメータの前に追加します。特定のまれな状況では、Key-Pair-Id の後にクエリ文字列パラメータを配置する必要があります。

   > **Important**
   >
   > パラメータに Policy、Signature、または Key-Pair-Id という名前を付けることはできません。

   独自のパラメータを追加する場合は、各パラメータの後に & を付加します (最後のパラメータにも付加します)。

4. **Policy=*########### base64 ##########***

   空白が削除されて base64 エンコードされた、JSON 形式のポリシーステートメント。詳細については、「カスタムポリシーを使用する署名付き URL のポリシーステートメントを作成する」を参照してください。

   ポリシーステートメントは、署名付き URL によってユーザーに許可されるアクセスをコントロールします。これには、ファイルの URL、有効期限切れ日時、URL が有効になる日時 (オプション)、ファイルへのアクセスが許可されている IP アドレスまたは IP アドレス範囲 (オプション) が含まれます。

5. **&Signature=*##############################***

   ハッシュ化され、署名された base64 エンコードバージョンの JSON ポリシーステートメント。詳細については、「カスタムポリシーを使用する署名付き URL の署名を作成する」を参照してください。

6. **&Key-Pair-Id=*########################## CloudFront ############### ID***

   CloudFront パブリックキーの ID (`K2JCJMDEHXQW5F` など)。パブリックキー ID は、署名付き URL の検証に使用するパブリックキーを CloudFront に通知します。CloudFront は、署名内の情報をポリシーステートメント内の情報と比較して、URL が改ざんされていないことを確認します。

   このパブリックキーは、ディストリビューションの信頼された署名者であるキーグループに属している必要があります。詳細については、「署名付き URL と署名付き Cookie を作成できる署名者を指定する」を参照してください。

カスタムポリシーを使用する署名付き URL のポリシーステートメントを作成する

カスタムポリシーを使用する署名付き URL のポリシーステートメントを作成するには、以下の手順を実行します。

さまざまな方法でファイルへのアクセスを制御するポリシーステートメントの例については、「the section called “カスタムポリシーを使用する署名付き URL のポリシーステートメントの例”」を参照してください。

カスタムポリシーを使用する署名付き URL のポリシーステートメントを作成するには

1. 以下の JSON 形式を使用してポリシーステートメントを構築します。小なり記号 (<) と大なり記号 (>)、およびそれらの中の説明は、独自の値に置き換えます。詳細については、「the

section called “カスタムポリシーを使用する署名付き URL のポリシーステートメントで指定する値”」を参照してください。

```
{
    "Statement": [
        {
            "Resource": "<Optional but recommended: URL of the file>",
            "Condition": {
                "DateLessThan": {
                 "AWS:EpochTime": <Required: ending date and time in Unix time
 format and UTC>
                },
                "DateGreaterThan": {
                 "AWS:EpochTime": <Optional: beginning date and time in Unix time
 format and UTC>
                },
                "IpAddress": {
                 "AWS:SourceIp": "<Optional: IP address>"
                }
            }
        }
    ]
}
```

次の点に注意してください。

- ポリシーには、1 つのステートメントのみを含めることができます。
- UTF-8 文字エンコードを使用します。
- すべての句読点およびパラメータ名を、指定されたとおりに正確に含めます。パラメータ名の省略形は受け付けられません。
- Condition セクションのパラメータの順序は問題ではありません。
- Resource、DateLessThan、DateGreaterThan、および IpAddress の値については、「the section called “カスタムポリシーを使用する署名付き URL のポリシーステートメントで指定する値”」を参照してください。

2. ポリシーステートメントからすべての空白 (タブと改行文字を含む) を削除します。アプリケーションコード内の文字列にエスケープ文字を含めることが必要になる場合があります。
3. MIME base64 エンコーディングを使用してポリシーステートメントを Base64 エンコードします。詳細については、RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One:

Format of Internet Message Bodies の Section 6.8, Base64 Content-Transfer-Encoding を参照してください。

4. URL クエリ文字列内の無効な文字を有効な文字で置き換えます。次の表に無効な文字と有効な文字を示します。

無効な文字 (置換元)	有効な文字 (置換先)
+	- (ハイフン)
=	_ (下線)
/	~ (チルダ)

5. 結果の値を署名付き URL の `Policy=` の後に付加します。
6. ポリシーステートメントのハッシュ化、署名、および base64 エンコードを行って、署名付き URL の署名を作成します。詳細については、「the section called “カスタムポリシーを使用する署名付き URL の署名を作成する”」を参照してください。

カスタムポリシーを使用する署名付き URL のポリシーステートメントで指定する値

カスタムポリシーのポリシーステートメントを作成する場合、以下の値を指定します。

リソース

クエリ文字列 (ある場合) が含まれる URL。ただし、CloudFront の `Policy`、`Signature`、および `Key-Pair-Id` パラメータが除外されます。例:

```
https://d111111abcdef8.cloudfront.net/images/horizon.jpg\?
size=large&license=yes
```

`Resource` の URL の値は 1 つだけ指定できます。

> **Important**
>
> ポリシーで、`Resource` パラメータを省略できますが、その場合、署名付き URL を持つすべてのユーザーが、署名付き URL の作成に使用するキーペアに関連付けられたあらゆるディストリビューションのすべてのファイルにアクセスできることになります。

次の点に注意してください。

- プロトコル – 値は http://、https://、または *:// で始まっている必要があります。
- クエリ文字列パラメータ – URL にクエリ文字列パラメータがある場合は、バックスラッシュ文字 (\) を使用してクエリ文字列の最初の疑問符 (?) をエスケープします。例:

  ```
  https://d111111abcdef8.cloudfront.net/images/horizon.jpg\?
  size=large&license=yes
  ```

- ワイルドカード文字 – ポリシーの URL にはワイルドカード文字を使用できます。次のワイルドカード文字がサポートされています。
  - アスタリスク (*) は、0 個以上の文字に一致します
  - 疑問符 (?) は、1 つの文字に一致します

  CloudFront がポリシー内の URL を HTTP リクエスト内の URL と照合すると、ポリシー内の URL は次のように 4 つのセクション (プロトコル、ドメイン、パス、クエリ文字列) に分割されます。

  ```
  [protocol]://[domain]/[path]\?[query string]
  ```

  ポリシーの URL にワイルドカード文字を使用する場合、ワイルドカードマッチングはワイルドカードを含むセクションの境界内でのみ適用されます。例えば、次のようなポリシーの URL を考えてみます。

  ```
  https://www.example.com/hello*world
  ```

  この例では、アスタリスクワイルドカード (*) はパスセクション内にのみ適用されるため、URL https://www.example.com/helloworld や https://www.example.com/hello-world には一致しますが、URL https://www.example.net/hello?world には一致しません。

  ワイルドカードマッチングのセクションの境界には、次の例外が適用されます。

  - パスセクションの末尾にアスタリスクがある場合、クエリ文字列セクションにアスタリスクが付いていることを意味します。たとえば、http://example.com/hello* と http://example.com/hello*\?* は同じです。
  - ドメインセクションの末尾にアスタリスクがある場合、パスセクションとクエリ文字列セクションの両方にアスタリスクが付いていることを意味します。たとえば、http://example.com* と http://example.com*/*\?* は同じです。

- ポリシーの URL では、プロトコルセクションを省略し、ドメインセクションをアスタリスクで始めることができます。その場合、プロトコルセクションは暗黙的にアスタリスクに設定されます。例えば、ポリシーの URL `*example.com` は `*://*example.com/` と同等です。
- アスタリスク (`"Resource": "*"`) 単独の場合は、どの URL にも一致します。

例えば、ポリシーの値 `https://d111111abcdef8.cloudfront.net/*game_download.zip*` は、次の URL すべてに一致します。

- `https://d111111abcdef8.cloudfront.net/game_download.zip`
- `https://d111111abcdef8.cloudfront.net/example_game_download.zip?license=yes`
- `https://d111111abcdef8.cloudfront.net/test_game_download.zip?license=temp`

- 代替ドメイン名 – ポリシーの URL で代替ドメイン名 (CNAME) を指定する場合、HTTP リクエストはウェブページまたはアプリケーション内の代替ドメイン名を使用する必要があります。ポリシーのファイルの Amazon S3 URL を指定しないでください。

DateLessThan

URL の有効期限切れ日時。Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。ポリシーでは、値を引用符で囲まないでください。UTC の詳細については、「Date and Time on the Internet: Timestamps」を参照してください。

たとえば、UTC の 2023 年 1 月 31 日午前 10 時 00 分は、Unix 時間形式の 1675159200 に変換されます。

これは、`Condition` セクションにおける唯一の必須パラメータです。CloudFront では、プライベートコンテンツへの永久的なアクセスがユーザーに許可されることのないよう、この値の指定が要求されます。

詳細については、「the section called “CloudFront が署名付き URL の有効期限切れの日時を確認するタイミング”」を参照してください。

DateGreaterThan (オプション)

オプションの URL 開始日時。Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。ユーザーは、指定された日時が過ぎるまでファイルにアクセスできません。値を引用符で囲まないでください。

IpAddress (オプション)

HTTP リクエストを実行するクライアントの IP アドレス。次の点に注意してください。

- ファイルへのアクセスをすべての IP アドレスに許可するには、`IpAddress` パラメータを省略します。
- IP アドレスまたは IP アドレス範囲を 1 つ指定できます。2 つの別々の範囲のどちらかにクライアントの IP アドレスが入っている場合にアクセスを許可するようなポリシーを使用することはできません。
- 1 つの IP アドレスからのアクセスを許可するには、以下のように指定します。

  `"IPv4 IP ####/32"`

- IP アドレス範囲は標準の IPv4 CIDR 形式 (`192.0.2.0/24` など) で指定する必要があります。詳細については、「Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan」を参照してください。

  > **Important**
  >
  > IPv6 形式の IP アドレス (例: 2001:0db8:85a3::8a2e:0370:7334) はサポートされていません。

  `IpAddress` を含むカスタムポリシーを使用する場合、ディストリビューションで IPv6 は有効にしません。一部のコンテンツへのアクセスを IP アドレスによって制限し、他のコンテンツで IPv6 リクエストをサポートする場合、2 つのディストリビューションを作成します。詳細については、トピック「the section called “ディストリビューションの設定”」の「the section called “IPv6 を有効にする”」を参照してください。

カスタムポリシーを使用する署名付き URL のポリシーステートメントの例

以下のポリシーステートメントの例は、特定のファイル、ディレクトリ内のすべてのファイル、またはキーペア ID に関連付けられたすべてのファイルへのアクセスを制御する方法を示しています。また、この例は、個々の IP アドレスまたは IP アドレス範囲からのアクセスを制御する方法、および指定された日時以降にユーザーが署名付き URL を使用することを禁止する方法も示しています。

これらの例のいずれかをコピーして貼り付ける場合は、すべての空白 (タブと改行文字を含む) を削除し、値を独自の値に置き換えて、右の中かっこ (}) の後に改行文字を含めます。

詳細については、「the section called “カスタムポリシーを使用する署名付き URL のポリシーステートメントで指定する値”」を参照してください。

トピック

- ポリシーステートメントの例: IP アドレス範囲から 1 つのファイルにアクセスする
- ポリシーステートメントの例: IP アドレス範囲からディレクトリ内のすべてのファイルにアクセスする
- ポリシーステートメントの例: キーペア ID に関連付けられたすべてのファイルに 1 つの IP アドレスからアクセスする

ポリシーステートメントの例: IP アドレス範囲から 1 つのファイルにアクセスする

次の署名付き URL 内のカスタムポリシーの例では、UTC の 2023 年 1 月 31 日午前 10 時 00 分まで、範囲 192.0.2.0/24 の IP アドレスから、ユーザーがファイル https://d111111abcdef8.cloudfront.net/game_download.zip にアクセスできることを指定しています。

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/game_download.zip",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.0/24"
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1675159200
                }
            }
        }
    ]
}
```

ポリシーステートメントの例: IP アドレス範囲からディレクトリ内のすべてのファイルにアクセスする

以下のカスタムポリシーの例では、Resource パラメータのアスタリスクワイルドカード文字 (*) が示すとおり、training ディレクトリ内のあらゆるファイルを対象とする署名付き URL を作成でき

ます。UTC の 2023 年 1 月 31 日午前 10 時 00 分まで、範囲 192.0.2.0/24 の IP アドレスから、ユーザーはファイルにアクセスできます。

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/training/*",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.0/24"
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1675159200
                }
            }
        }
    ]
}
```

このポリシーを使用する各署名付き URL には、次のように、特定のファイルを識別する URL があります。

https://d111111abcdef8.cloudfront.net/training/orientation.pdf

ポリシーステートメントの例: キーペア ID に関連付けられたすべてのファイルに 1 つの IP アドレスからアクセスする

以下のカスタムポリシーの例では、Resource パラメータのアスタリスクワイルドカード文字 (*) が示すとおり、あらゆるディストリビューションに関連付けられたあらゆるファイルを対象とする署名付き URL を作成できます。署名付き URL には、http:// ではなく https:// プロトコルを使用する必要があります。ユーザーは IP アドレス 192.0.2.10/32 を使用する必要があります。(CIDR 表記の値 192.0.2.10/32 は 1 つの IP アドレス 192.0.2.10 を参照します)。ファイルは、UTC の 2023 年 1 月 31 日午前 10 時 00 分から UTC の 2023 年 2 月 2 日午前 10 時 00 分まで使用できます。

```
{
    "Statement": [
       {
            "Resource": "https://*",
            "Condition": {
                "IpAddress": {
```

```
                    "AWS:SourceIp": "192.0.2.10/32"
                },
                "DateGreaterThan": {
                    "AWS:EpochTime": 1675159200
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1675332000
                }
            }
        }
    ]
}
```

このポリシーを使用する各署名付き URL には、次のように、特定の CloudFront ディストリビューション内の特定のファイルを識別する URL があります。

```
https://d111111abcdef8.cloudfront.net/training/orientation.pdf
```

署名付き URL にはキーペア ID も含まれます。キーペア ID は、URL に指定されたディストリビューション (d111111abcdef8.cloudfront.net) 内の信頼されたキーグループに関連付けられる必要があります。

カスタムポリシーを使用する署名付き URL の署名を作成する

カスタムポリシーを使用する署名付き URL の署名は、ハッシュ化、署名、および base64 エンコードが行われたバージョンのポリシーステートメントです。カスタムポリシーの署名を作成するには、以下の手順を実行します。

ポリシーステートメントのハッシュ化、署名、およびエンコードを行う方法の詳細および例については、以下の各資料を参照してください。

- Linux コマンドおよび OpenSSL を使用した Base64 エンコードおよび暗号化
- 署名付き URL の署名を作成するためのコード例

オプション 1: カスタムポリシーを使用して署名を作成するには

1. 「カスタムポリシーを使用する署名付き URL のポリシーステートメントを作成するには」の手順で作成した JSON ポリシーステートメントを、SHA-1 ハッシュ関数と RSA を使用してハッシュ化し、署名します。空白は含まれていないが、まだ base64 エンコードされていないバージョンのポリシーステートメントを使用します。

ハッシュ関数に必要なプライベートキーには、対応するパブリックキーがディストリビューション内のアクティブな信頼されたキーグループにあるものを使用してください。

> **Note**
>
> ポリシーステートメントをハッシュ化および署名するための方法は、プログラミング言語およびプラットフォームによって異なります。サンプルコードについては、「署名付き URL の署名を作成するためのコード例」を参照してください。

2. ハッシュ化および署名された文字列から、空白 (タブや改行文字を含む) を削除します。
3. MIME base64 エンコーディングを使用して文字列を Base64 エンコードします。詳細については、RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies の Section 6.8, Base64 Content-Transfer-Encoding を参照してください。
4. URL クエリ文字列内の無効な文字を有効な文字で置き換えます。次の表に無効な文字と有効な文字を示します。

無効な文字 (置換元)	有効な文字 (置換先)
+	- (ハイフン)
=	_ (下線)
/	~ (チルダ)

5. 結果の値を署名付き URL の `&Signature=` の後に付加し、「カスタムポリシーを使用して署名付き URL を作成するには」に戻って、署名付き URL の各パートの連結を終了します。

署名付き Cookie を使用する

CloudFront 署名付き Cookie を使用すると、現在の URL を変更したくない場合や、複数の制限付きファイル (ウェブサイトの購読者の領域にあるすべてのファイルなど) へのアクセスを提供する場合に、誰がコンテンツにアクセスできるかを制御できます。このトピックでは、署名付き Cookie を使用する際の考慮事項と、既定ポリシーとカスタムポリシーを使用するように署名付き Cookie を設定する方法について説明します。

トピック

- 署名付き Cookie に既定ポリシーを使用するか、カスタムポリシーを使用するかを決定する

- 署名付き Cookie の仕組み
- 署名付き Cookie の悪用を防止する
- CloudFront が署名付き Cookie の有効期限切れ日時を確認するタイミング
- サンプルコードおよびサードパーティーツール
- 既定ポリシーを使用して署名付き Cookie を設定する
- カスタムポリシーを使用して署名付き Cookie を設定する

署名付き Cookie に既定ポリシーを使用するか、カスタムポリシーを使用するかを決定する

署名付き Cookie を作成する場合、Cookie の有効期間など、署名付き Cookie で制限を指定する JSON 形式のポリシーステートメントを作成します。既定ポリシーまたはカスタムポリシーを使用できます。次の表では、既定ポリシーとカスタムポリシーを比較しています。

説明	既定ポリシー	カスタムポリシー
ポリシーステートメントを複数のファイル用に再利用できる。ポリシーステートメントを再利用するには、`Resource` オブジェクトでワイルドカード文字を使用する必要があります。(詳しくは、「署名付き Cookie のカスタムポリシーのポリシーステートメントで指定する値」を参照してください)。	いいえ	はい
ユーザーがコンテンツへのアクセスを開始できる日時を指定できる	いいえ	はい (オプション)
ユーザーがコンテンツにアクセスできなくなる日時を指定できる	はい	はい
コンテンツにアクセスできるユーザーの IP アドレスまたは IP アドレス範囲を指定できる	いいえ	はい (オプション)

既定ポリシーを使用して署名付き Cookie を作成する方法については、「既定ポリシーを使用して署名付き Cookie を設定する」を参照してください。

カスタムポリシーを使用して署名付き Cookie を作成する方法については、「カスタムポリシーを使用して署名付き Cookie を設定する」を参照してください。

署名付き Cookie の仕組み

ここでは、署名付き Cookie 用に CloudFront を設定する方法と、ユーザーが署名付き Cookie を含むリクエストを送信した場合の CloudFront の応答の概要を示します。

1. CloudFront ディストリビューションで、1 つ以上の信頼されたキーグループを指定します。指定したグループには、CloudFront が URL 署名の検証に使用できるパブリックキーが含まれている必要があります。対応するプライベートキーを使用して URL に署名します。

   詳細については、「署名付き URL と署名付き Cookie を作成できる署名者を指定する」を参照してください。
2. ユーザーがコンテンツにアクセスできるかどうかを判断し、アクセスできる場合は、3 つの `Set-Cookie` ヘッダーをビューワーに送信するアプリケーションを開発します (各 `Set-Cookie` ヘッダーには名前と値のペアを 1 つだけ含めることができ、CloudFront 署名付き Cookie では 3 つの名前と値のペアが必要です)。ビューワーがプライベートコンテンツをリクエストする前に、ビューワーに `Set-Cookie` ヘッダーを送信する必要があります。Cookie の有効期限を短く設定した場合、ユーザーがアクセスを続行できるように、以降のリクエストに対してさらに 3 つの `Set-Cookie` ヘッダーを送信することもできます。

   通常、CloudFront ディストリビューションには少なくとも 2 つのキャッシュ動作があります。認証を必要としないものと、認証を必要とするものです。サイトのセキュリティで保護された部分のエラーページには、ログインページへのリダイレクタまたはリンクが含まれます。

   Cookie に基づいてファイルをキャッシュするようにディストリビューションを設定している場合、CloudFront は署名付き Cookie の属性に基づいて個別のファイルをキャッシュしません。
3. ユーザーがウェブサイトにサインインし、コンテンツに対して支払いをするか、またはその他のアクセスの要件を満たします。
4. アプリケーションは、レスポンスで `Set-Cookie` ヘッダーを返し、ビューワーは名前と値のペアを格納します。
5. ユーザーがファイルをリクエストします。

   ユーザーのブラウザまたはその他のビューワーは、ステップ 4 の名前と値のペアを取得し、リクエストの `Cookie` ヘッダーに追加します。これが署名付き Cookie です。
6. CloudFront はパブリックキーを使用して、署名付き Cookie の署名を検証し、Cookie が改ざんされていないことを確認します。署名が無効である場合、リクエストは拒否されます。

Cookie の署名が有効である場合、CloudFront は Cookie のポリシーステートメントを参照して(または既定ポリシーを使用している場合はポリシーステートメントを作成して)、リクエストがまだ有効であることを確認します。たとえば、Cookie の開始日時と終了日時が指定されていれば、CloudFront は、アクセスが許可されている期間にユーザーがコンテンツへのアクセスを試みていることを確認します。

リクエストがポリシーステートメントの要件を満たしている場合、CloudFront は制限されていないコンテンツの場合と同様にコンテンツを供給します。つまり、ファイルがエッジキャッシュにすでに存在するかどうかを確認し、必要に応じてリクエストをオリジンに転送して、ファイルをユーザーに返します。

署名付き Cookie の悪用を防止する

`Set-Cookie` ヘッダーで `Domain` パラメータを指定する場合、同一ルートドメイン名を使用するユーザーによる潜在的なアクセスを低減できる、最も厳密な値を指定します。たとえば、apex.example.com は、特に example.com を制御しない場合は、example.com よりも優先されます。これによって、ユーザーが example.com のコンテンツにアクセスすることを防止できます。

この種類の攻撃を防ぐには、以下の作業を行います。

- `Expires` ヘッダーでセッション Cookie が作成されるように、`Max-Age` および `Set-Cookie` Cookie 属性を除外します。セッション Cookie は、ユーザーがブラウザを閉じたときに自動的に削除されるため、ユーザーがコンテンツに不正アクセスする可能性が低くなります。
- ビューワーがリクエストに Cookie を含める場合に Cookie が暗号化されるように、`Secure` 属性を含めます。
- 可能な場合、カスタムポリシーを使用してビューワーの IP アドレスを含めます。
- `CloudFront-Expires` 属性では、ユーザーがコンテンツにアクセスできるようにする期間に基づいて、最短で適切な有効期限の時刻を指定します。

CloudFront が署名付き Cookie の有効期限切れ日時を確認するタイミング

署名付き Cookie がまだ有効であるかどうかを確認するために、CloudFront は HTTP リクエスト時に、Cookie の有効期限切れ日時を確認します。有効期限切れ時刻の直前にクライアントが大きなファイルのダウンロードを開始した場合、ダウンロード中に有効期限切れ時刻が経過してもダウンロードは完了します。TCP 接続が中断し、有効期限切れ時刻が経過した後にクライアントがダウンロードを再開した場合、ダウンロードは失敗します。

クライアントが、ファイルを断片的に取得するレンジ GET を使用した場合、有効期限切れ時刻が経過した後に実行された GET リクエストは失敗します。レンジ GET の詳細については、「CloudFront がオブジェクトの部分的リクエスト (レンジ GET) を処理する方法」を参照してください。

サンプルコードおよびサードパーティーツール

プライベートコンテンツ用のサンプルコードは、署名付き URL の署名を作成する方法のみを示しています。ただし、署名付き Cookie の署名を作成するプロセスは非常によく似ており、サンプルコードの多くの部分は関連しています。詳細については、以下のトピックを参照してください。

- Perl を使用して URL 署名を作成する
- PHP を使用して URL 署名を作成する
- C# と .NET Framework を使用して URL 署名を作成する
- Java を使用して URL 署名を作成する

既定ポリシーを使用して署名付き Cookie を設定する

既定ポリシーを使用して署名付き Cookie を設定するには、以下のステップを実行します。署名を作成するには、「既定ポリシーを使用する署名付き Cookie の署名を作成する」を参照してください。

既定ポリシーを使用して署名付き Cookie を設定するには

1. .NET または Java を使用して署名付き Cookie を作成しており、キーペアのプライベートキーをデフォルトの .pem 形式から .NET または Java 対応の形式に変更していない場合は、それを変換します。詳細については、「プライベートキーの形式を変更する (.NET および Java のみ)」を参照してください。
2. 承認されたビューワーに 3 つの `Set-Cookie` ヘッダーを送信するアプリケーションをプログラムします。各 `Set-Cookie` ヘッダーには名前と値のペアを 1 つだけ含めることができ、CloudFront 署名付き Cookie では 3 つの名前と値のペアが必要であるため、3 つの `Set-Cookie` ヘッダーが必要です。名前と値のペアは、`CloudFront-Expires`、`CloudFront-Signature`、および `CloudFront-Key-Pair-Id` です。アクセスを制御するファイルに対してユーザーが最初のリクエストを行う前に、値がビューワーに存在している必要があります。

   > **Note**
   >
   > 一般的に、`Expires` 属性と `Max-Age` 属性を除外することをお勧めします。これらの属性を除外すると、ユーザーがブラウザを閉じたときに、ブラウザで Cookie が削除され

るため、ユーザーがコンテンツに不正アクセスする可能性が低くなります。詳細については、「署名付き Cookie の悪用を防止する」を参照してください。

Cookie の属性の名前では、大文字と小文字が区別されます。

改行は、属性を判読しやすくするためにのみ含まれています。

```
Set-Cookie:
CloudFront-Expires=date and time in Unix time format (in seconds) and Coordinated
 Universal Time (UTC);
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly

Set-Cookie:
CloudFront-Signature=hashed and signed version of the policy statement;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly

Set-Cookie:
CloudFront-Key-Pair-Id=public key ID for the CloudFront public key whose
 corresponding private key you're using to generate the signature;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly
```

(オプション) **Domain**

リクエストされたファイルのドメイン名。Domain 属性を指定しない場合、デフォルト値は URL のドメイン名で、指定されたドメイン名にのみ適用され、サブドメインには適用されません。Domain 属性を指定する場合、サブドメインにも適用されます。ドメイン名の先頭のドット (例えば、Domain=.example.com) はオプションです。さらに、Domain 属性を指定する場合は、URL のドメイン名と Domain 属性の値が一致している必要があります。

CloudFront がディストリビューションに割り当てたドメイン名 (d111111abcdef8.cloudfront.net など) を指定することはできますが、*.cloudfront.net をドメイン名として指定することはできません。

URL で代替ドメイン名 (example.com など) を使用する場合は、Domain 属性を指定するかどうかにかかわらず、代替ドメイン名をディストリビューションに追加する必要があります。詳細については、トピック「ディストリビューション設定リファレンス」の「代替ドメイン名 (CNAME)」を参照してください。

(オプション) **Path**

リクエストされたファイルのパス。Path 属性を指定しない場合、デフォルト値は URL のパスです。

Secure

リクエストを送信する前に、ビューワーが Cookie を暗号化することを要求します。Cookie の属性を中間者攻撃から保護するために、HTTPS 接続で Set-Cookie ヘッダーを送信することをお勧めします。

HttpOnly

ブラウザ (サポートされている場合) が Cookie 値とどのようにやり取りするかを定義します。HttpOnly では、Cookie 値は JavaScript からアクセスできません。この予防策は、クロスサイトスクリプティング (XSS) 攻撃の軽減に役立ちます。詳細については、「HTTP Cookie の使用」を参照してください。

CloudFront-Expires

有効期限切れ日時を Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。たとえば、UTC の 2013 年 1 月 1 日午前 10 時 00 分は、Unix 時間形式の 1357034400 に変換されます。エポック時間を使用するには、日付に 2147483647 (2038 年 1 月 19 日 03:14:07 UTC) より前の 32 ビット整数を使用します。UTC の詳細については、RFC 3339, Date and Time on the Internet: Timestamps (https://tools.ietf.org/html/rfc3339) を参照してください。

CloudFront-Signature

ハッシュ化され、署名された base64 エンコードバージョンの JSON ポリシーステートメント。詳細については、「既定ポリシーを使用する署名付き Cookie の署名を作成する」を参照してください。

CloudFront-Key-Pair-Id

CloudFront パブリックキーの ID (K2JCJMDEHXQW5F など)。パブリックキー ID は、署名付き URL の検証に使用するパブリックキーを CloudFront に通知します。CloudFront は、署名内の情報をポリシーステートメント内の情報と比較して、URL が改ざんされていないことを確認します。

このパブリックキーは、ディストリビューションの信頼された署名者であるキーグループに属している必要があります。詳細については、「署名付き URL と署名付き Cookie を作成できる署名者を指定する」を参照してください。

以下は、ファイルの URL のディストリビューションに関連付けられたドメイン名を使用する場合の、1 つの署名付き Cookie の Set-Cookie ヘッダーの例です。

```
Set-Cookie: CloudFront-Expires=1426500000; Domain=d111111abcdef8.cloudfront.net; Path=/
images/*; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=yXrSIgyQoeE4FBI4eMKF6ho~CA8_;
 Domain=d111111abcdef8.cloudfront.net; Path=/images/*; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F;
 Domain=d111111abcdef8.cloudfront.net; Path=/images/*; Secure; HttpOnly
```

以下は、ファイルの URL に代替ドメイン名 example.org を使用している場合の、1 つの署名付き Cookie の Set-Cookie ヘッダーの例です。

```
Set-Cookie: CloudFront-Expires=1426500000; Domain=example.org; Path=/images/*; Secure;
 HttpOnly
Set-Cookie: CloudFront-Signature=yXrSIgyQoeE4FBI4eMKF6ho~CA8_; Domain=example.org;
 Path=/images/*; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F; Domain=example.org; Path=/images/*;
 Secure; HttpOnly
```

URL で代替ドメイン名 (example.com など) を使用する場合は、Domain 属性を指定するかどうかにかかわらず、代替ドメイン名をディストリビューションに追加する必要があります。詳細については、トピック「ディストリビューション設定リファレンス」の「代替ドメイン名 (CNAME)」を参照してください。

既定ポリシーを使用する署名付き Cookie の署名を作成する

既定ポリシーを使用する署名付き Cookie の署名を作成するには、次の手順を実行します。

トピック

- 既定ポリシーを使用する署名付き Cookie のポリシーステートメントを作成する
- 既定ポリシーを使用する署名付き Cookie の署名を作成するためのポリシーステートメントに署名する

既定ポリシーを使用する署名付き Cookie のポリシーステートメントを作成する

既定ポリシーを使用する署名付き Cookie を設定した場合、CloudFront-Signature 属性は、ポリシーステートメントのハッシュ化および署名されたバージョンです。カスタムポリシーを使用する署名付き Cookie とは異なり、既定ポリシーを使用する署名付き Cookie では、Set-Cookie ヘッダーにポリシーステートメントを含めません。ポリシーステートメントを作成するには、以下の手順を実行します。

既定ポリシーを使用する署名付き Cookie のポリシーステートメントを作成するには

1. 以下の JSON 形式および UTF-8 文字エンコードを使用してポリシーステートメントを構築します。すべての句読点および他のリテラル値を、指定されたとおりに正確に含めます。Resource および DateLessThan パラメータの詳細については、「署名付き Cookie の既定ポリシーのポリシーステートメントで指定する値」を参照してください。

```
{
    "Statement": [
        {
            "Resource": "base URL or stream name",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime": ending date and time in Unix time format and
 UTC
                }
            }
        }
    ]
}
```

2. ポリシーステートメントからすべての空白 (タブと改行文字を含む) を削除します。アプリケーションコード内の文字列にエスケープ文字を含めることが必要になる場合があります。

署名付き Cookie の既定ポリシーのポリシーステートメントで指定する値

既定ポリシーのポリシーステートメントを作成する場合、以下の値を指定します。

リソース

クエリ文字列 (存在する場合) を含むベース URL。以下に例を示します。

```
https://d111111abcdef8.cloudfront.net/images/horizon.jpg?
size=large&license=yes
```

Resource の日付形式は 1 つだけ指定できます。

次の点に注意してください。

- プロトコル – 値は http:// または https:// で始まっている必要があります。
- クエリ文字列パラメータ – クエリ文字列パラメータがない場合は、疑問符を省略します。
- 代替ドメイン名 – URL で代替ドメイン名 (CNAME) を指定する場合は、ウェブページまたはアプリケーション内のファイルを参照するときに代替ドメイン名を指定する必要があります。ファイルの Amazon S3 URL を指定しないでください。

DateLessThan

URL の有効期限切れ日時。Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。値を引用符で囲まないでください。

たとえば、UTC の 2015 年 3 月 16 日午前 10 時 00 分は、UNIX 時間形式の 1426500000 に変換されます。

この値は、CloudFront-Expires ヘッダーの Set-Cookie 属性の値と一致する必要があります。値を引用符で囲まないでください。

詳細については、「CloudFront が署名付き Cookie の有効期限切れ日時を確認するタイミング」を参照してください。

既定ポリシーのポリシーステートメントの例

署名付き Cookie 内で以下のポリシーステートメントの例を使用すると、ユーザーは、UTC の 2015 年 3 月 16 日午前 10 時 00 分までファイル https://d111111abcdef8.cloudfront.net/horizon.jpg にアクセスできます。

```
{
```

```
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/horizon.jpg?
size=large&license=yes",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime": 1426500000
                }
            }
        }
    ]
}
```

既定ポリシーを使用する署名付き Cookie の署名を作成するためのポリシーステートメントに署名する

CloudFront-Signature ヘッダーの Set-Cookie 属性の値を作成するには、「既定ポリシーを使用する署名付き Cookie のポリシーステートメントを作成するには」で作成したポリシーステートメントをハッシュ化して署名します。

ポリシーステートメントのハッシュ化、署名、およびエンコードを行う方法の詳細および例については、以下のトピックを参照してください。

- Linux コマンドおよび OpenSSL を使用した Base64 エンコードおよび暗号化
- 署名付き URL の署名を作成するためのコード例

既定ポリシーを使用して署名付き Cookie の署名を作成するには

1. 「既定ポリシーを使用する署名付き Cookie のポリシーステートメントを作成するには」の手順で作成したポリシーステートメントを、SHA-1 ハッシュ関数と RSA を使用してハッシュ化し、署名します。空白を含まないバージョンのポリシーステートメントを使用します。

   ハッシュ関数に必要なプライベートキーには、対応するパブリックキーがディストリビューション内のアクティブな信頼されたキーグループにあるものを使用してください。

   > **Note**
   >
   > ポリシーステートメントをハッシュ化および署名するための方法は、プログラミング言語およびプラットフォームによって異なります。サンプルコードについては、「署名付き URL の署名を作成するためのコード例」を参照してください。

2. ハッシュ化および署名された文字列から、空白 (タブや改行文字を含む) を削除します。
3. MIME base64 エンコーディングを使用して文字列を Base64 エンコードします。詳細については、RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies の Section 6.8, Base64 Content-Transfer-Encoding を参照してください。
4. URL クエリ文字列内の無効な文字を有効な文字で置き換えます。次の表に無効な文字と有効な文字を示します。

無効な文字 (置換元)	有効な文字 (置換先)
+	- (ハイフン)
=	_ (下線)
/	~ (チルダ)

5. 結果の値を、`Set-Cookie` の名前と値のペアの `CloudFront-Signature` ヘッダーに含めます。次に、「既定ポリシーを使用して署名付き Cookie を設定するには」に戻り、`Set-Cookie` の `CloudFront-Key-Pair-Id` ヘッダーを追加します。

カスタムポリシーを使用して署名付き Cookie を設定する

カスタムポリシーを使用する署名付き Cookie を設定するには、以下の手順を実行します。

カスタムポリシーを使用して署名付き Cookie を設定するには

1. .NET または Java を使用して署名付き URL を作成しており、キーペアのプライベートキーをデフォルトの .pem 形式から .NET または Java 対応の形式に変更していない場合は、それを変換します。詳細については、「プライベートキーの形式を変更する (.NET および Java のみ)」を参照してください。
2. 承認されたビューワーに 3 つの `Set-Cookie` ヘッダーを送信するアプリケーションをプログラムします。各 `Set-Cookie` ヘッダーには名前と値のペアを 1 つだけ含めることができ、CloudFront 署名付き Cookie では 3 つの名前と値のペアが必要であるため、3 つの `Set-Cookie` ヘッダーが必要です。名前と値のペアは、`CloudFront-Policy`、`CloudFront-Signature`、および `CloudFront-Key-Pair-Id` です。アクセスを制御するファイルに対してユーザーが最初のリクエストを行う前に、値がビューワーに存在している必要があります。

> **Note**
>
> 一般的に、Expires 属性と Max-Age 属性を除外することをお勧めします。これにより、ユーザーがブラウザを閉じたときに、ブラウザで Cookie が削除されるため、ユーザーがコンテンツに不正アクセスする可能性が低くなります。詳細については、「署名付き Cookie の悪用を防止する」を参照してください。

Cookie の属性の名前では、大文字と小文字が区別されます。

改行は、属性を判読しやすくするためにのみ含まれています。

```
Set-Cookie:
CloudFront-Policy=base64 encoded version of the policy statement;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly


Set-Cookie:
CloudFront-Signature=hashed and signed version of the policy statement;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly

Set-Cookie:
CloudFront-Key-Pair-Id=public key ID for the CloudFront public key whose
 corresponding private key you're using to generate the signature;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly
```

(オプション) **Domain**

リクエストされたファイルのドメイン名。Domain 属性を指定しない場合、デフォルト値は URL のドメイン名で、指定されたドメイン名にのみ適用され、サブドメインには適用されません。Domain 属性を指定する場合、サブドメインにも適用されます。ドメイン名の先頭の

ドット (例えば、Domain=.example.com) はオプションです。さらに、Domain 属性を指定する場合は、URL のドメイン名と Domain 属性の値が一致している必要があります。

CloudFront がディストリビューションに割り当てたドメイン名 (d111111abcdef8.cloudfront.net など) を指定することはできますが、*.cloudfront.net をドメイン名として指定することはできません。

URL で代替ドメイン名 (example.com など) を使用する場合は、Domain 属性を指定するかどうかにかかわらず、代替ドメイン名をディストリビューションに追加する必要があります。詳細については、トピック「ディストリビューション設定リファレンス」の「代替ドメイン名 (CNAME)」を参照してください。

(オプション) **Path**

リクエストされたファイルのパス。Path 属性を指定しない場合、デフォルト値は URL のパスです。

Secure

リクエストを送信する前に、ビューワーが Cookie を暗号化することを要求します。Cookie の属性を中間者攻撃から保護するために、HTTPS 接続で Set-Cookie ヘッダーを送信することをお勧めします。

HttpOnly

ビューワーが HTTP または HTTPS リクエストでのみ Cookie を送信することを要求します。

CloudFront-Policy

空白が削除されて base64 エンコードされた、JSON 形式のポリシーステートメント。詳細については、「カスタムポリシーを使用する署名付き Cookie の署名を作成する」を参照してください。

ポリシーステートメントは、署名付き Cookie によってユーザーに許可されるアクセスをコントロールします。これには、ユーザーがアクセスできるファイル、有効期限切れ日時、URL が有効になる日時 (オプション)、ファイルへのアクセスが許可されている IP アドレスまたは IP アドレス範囲 (オプション) が含まれます。

CloudFront-Signature

ハッシュ化され、署名された base64 エンコードバージョンの JSON ポリシーステートメント。詳細については、「カスタムポリシーを使用する署名付き Cookie の署名を作成する」を参照してください。

CloudFront-Key-Pair-Id

CloudFront パブリックキーの ID (K2JCJMDEHXQW5F など)。パブリックキー ID は、署名付き URL の検証に使用するパブリックキーを CloudFront に通知します。CloudFront は、署名内の情報をポリシーステートメント内の情報と比較して、URL が改ざんされていないことを確認します。

このパブリックキーは、ディストリビューションの信頼された署名者であるキーグループに属している必要があります。詳細については、「署名付き URL と署名付き Cookie を作成できる署名者を指定する」を参照してください。

カスタムポリシーの **Set-Cookie** ヘッダーの例

以下の Set-Cookie ヘッダーペアの例を参照してください。

代替ドメイン名 (example.org など) を URL で使用する場合は、Domain 属性を指定するかどうかにかかわらず、代替ドメイン名をディストリビューションに追加する必要があります。詳細については、トピック「ディストリビューション設定リファレンス」の「代替ドメイン名 (CNAME)」を参照してください。

Example 例 1

ディストリビューションに関連するドメイン名をファイルの URL で使用している場合、1 つの署名付き Cookie で Set-Cookie ヘッダーを使用できます。

```
Set-Cookie: CloudFront-
Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2QxMTExMTFhYmNkZWY4LmNsb3VkZnJvbnQubmV0L2dh
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_;
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F;
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
```

Example 例 2

代替ドメイン名 (example.org) をファイルの URL で使用している場合、1 つの署名付き Cookie で Set-Cookie ヘッダーを使用できます。

```
Set-Cookie: CloudFront-
Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2QxMTExMTFhYmNkZWY4LmNsb3VkZnJvbnQubmV0L2dh
 Domain=example.org; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_; Domain=example.org;
 Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F; Domain=example.org; Path=/; Secure;
 HttpOnly
```

Example 例 3

ディストリビューションに関連するドメイン名をファイルの URL で使用している場合、1 つの署名付きリクエストで Set-Cookie ヘッダーペアを使用できます。

```
Set-Cookie: CloudFront-
Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2QxMTExMTFhYmNkZWY4LmNsb3VkZnJvbnQubmV0L2dh
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_;
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F;
 Domain=dd111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
```

Example 例 4

ディストリビューションに関連付けられた代替ドメイン名をファイルの URL で使用している場合、1 つの署名付きリクエストで Set-Cookie ヘッダーペアを使用できます。

```
Set-Cookie: CloudFront-
Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2QxMTExMTFhYmNkZWY4LmNsb3VkZnJvbnQubmV0L2dh
 Domain=example.org; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_; Domain=example.org;
 Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F; Domain=example.org; Path=/; Secure;
 HttpOnly
```

既定ポリシーを使用する署名付き Cookie のポリシーステートメントを作成する

カスタムポリシーのポリシーステートメントを作成するには、以下の手順を実行します。さまざまな方法でファイルへのアクセスを制御するポリシーステートメントのいくつかの例については、「カスタムポリシーを使用する署名付き Cookie のポリシーステートメントの例」を参照してください。

カスタムポリシーを使用する署名付き Cookie のポリシーステートメントを作成するには

1. 以下の JSON 形式を使用してポリシーステートメントを構築します。

```
{
    "Statement": [
        {
            "Resource": "URL of the file",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime":required ending date and time in Unix time
 format and UTC
                },
                "DateGreaterThan": {
                    "AWS:EpochTime":optional beginning date and time in Unix time
 format and UTC
                },
                "IpAddress": {
                    "AWS:SourceIp": "optional IP address"
                }
            }
        }
    ]
}
```

次の点に注意してください。

- 1 つのステートメントのみを含めることができます。
- UTF-8 文字エンコードを使用します。
- すべての句読点およびパラメータ名を、指定されたとおりに正確に含めます。パラメータ名の省略形は受け付けられません。
- Condition セクションのパラメータの順序は問題ではありません。

- Resource、DateLessThan、DateGreaterThan、および IpAddress の値については、「署名付き Cookie のカスタムポリシーのポリシーステートメントで指定する値」を参照してください。

2. ポリシーステートメントからすべての空白 (タブと改行文字を含む) を削除します。アプリケーションコード内の文字列にエスケープ文字を含めることが必要になる場合があります。
3. MIME base64 エンコーディングを使用してポリシーステートメントを Base64 エンコードします。詳細については、RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies の Section 6.8, Base64 Content-Transfer-Encoding を参照してください。
4. URL クエリ文字列内の無効な文字を有効な文字で置き換えます。次の表に無効な文字と有効な文字を示します。

無効な文字 (置換元)	有効な文字 (置換先)
+	- (ハイフン)
=	_ (下線)
/	~ (チルダ)

5. 結果の値を、Set-Cookie ヘッダーの CloudFront-Policy= の後に含めます。
6. ポリシーステートメントのハッシュ化、署名、および base64 エンコードを行って、Set-Cookie 用に CloudFront-Signature ヘッダーの署名を作成します。詳細については、「カスタムポリシーを使用する署名付き Cookie の署名を作成する」を参照してください。

署名付き Cookie のカスタムポリシーのポリシーステートメントで指定する値

カスタムポリシーのポリシーステートメントを作成する場合、以下の値を指定します。

リソース

クエリ文字列 (存在する場合) を含むベース URL。

```
https://d111111abcdef8.cloudfront.net/images/horizon.jpg?
size=large&license=yes
```

> **Important**
>
> Resource パラメータを省略した場合、ユーザーは、署名付き URL の作成に使用するキーペアに関連付けられたあらゆるディストリビューションに関連付けられるすべてのファイルにアクセスできます。

Resource の日付形式は 1 つだけ指定できます。

次の点に注意してください。

- プロトコル – 値は http:// または https:// で始まっている必要があります。
- クエリ文字列パラメータ – クエリ文字列パラメータがない場合は、疑問符を省略します。
- ワイルドカード – 0 個以上の文字に一致するワイルドカード文字 (*)、または 1 つの文字に一致するワイルドカード文字 (?) を使用できます。文字列のどこにでも含めることができます。次に例を示します。この値は、

  ```
  https://d111111abcdef8.cloudfront.net/*game_download.zip*
  ```

  たとえば、次のファイルを含みます。

  - https://d111111abcdef8.cloudfront.net/game_download.zip
  - https://d111111abcdef8.cloudfront.net/example_game_download.zip?license=yes
  - https://d111111abcdef8.cloudfront.net/test_game_download.zip?license=temp
- 代替ドメイン名 – URL で代替ドメイン名 (CNAME) を指定する場合は、ウェブページまたはアプリケーション内のファイルを参照するときに代替ドメイン名を指定する必要があります。ファイルの Amazon S3 URL を指定しないでください。

DateLessThan

URL の有効期限切れ日時。Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。値を引用符で囲まないでください。

たとえば、UTC の 2015 年 3 月 16 日午前 10 時 00 分は、UNIX 時間形式の 1426500000 に変換されます。

詳細については、「CloudFront が署名付き Cookie の有効期限切れ日時を確認するタイミング」を参照してください。

DateGreaterThan (オプション)

オプションの URL 開始日時。Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。ユーザーは、指定された日時が過ぎるまでファイルにアクセスできません。値を引用符で囲まないでください。

IpAddress (オプション)

GET リクエストを実行するクライアントの IP アドレス。次の点に注意してください。

- ファイルへのアクセスをすべての IP アドレスに許可するには、`IpAddress` パラメータを省略します。
- IP アドレスまたは IP アドレス範囲を 1 つ指定できます。たとえば、2 つの別々の範囲のどちらかにクライアントの IP アドレスが入っている場合にアクセスを許可するようなポリシーを設定することはできません。
- 1 つの IP アドレスからのアクセスを許可するには、以下のように指定します。

  `"IPv4 IP ####/32"`
- IP アドレス範囲は標準の IPv4 CIDR 形式 (`192.0.2.0/24` など) で指定する必要があります。詳細については、RFC 4632, Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan (https://tools.ietf.org/html/rfc4632) を参照してください。

> **Important**
>
> IPv6 形式の IP アドレス (例: 2001:0db8:85a3::8a2e:0370:7334) はサポートされていません。

`IpAddress` を含むカスタムポリシーを使用する場合、ディストリビューションで IPv6 は有効にしません。一部のコンテンツへのアクセスを IP アドレスによって制限し、他のコンテンツで IPv6 リクエストをサポートする場合、2 つのディストリビューションを作成します。詳細については、トピック「ディストリビューション設定リファレンス」の「IPv6 を有効にする」を参照してください。

カスタムポリシーを使用する署名付き Cookie のポリシーステートメントの例

以下のポリシーステートメントの例は、特定のファイル、ディレクトリ内のすべてのファイル、またはキーペア ID に関連付けられたすべてのファイルへのアクセスを制御する方法を示しています。ま

た、この例は、個々の IP アドレスまたは IP アドレス範囲からのアクセスを制御する方法、および指定された日時以降にユーザーが署名付き Cookie を使用することを禁止する方法も示しています。

これらの例のいずれかをコピーして貼り付ける場合は、すべての空白 (タブと改行文字を含む) を削除し、値を独自の値で置き換え、右の中かっこ (}) の後に改行文字を含めます。

詳細については、「署名付き Cookie のカスタムポリシーのポリシーステートメントで指定する値」を参照してください。

トピック

- ポリシーステートメントの例: IP アドレス範囲から 1 つのファイルにアクセスする
- ポリシーステートメントの例: IP アドレス範囲からディレクトリ内のすべてのファイルにアクセスする
- ポリシーステートメントの例: キーペア ID に関連付けられたすべてのファイルに 1 つの IP アドレスからアクセスする

ポリシーステートメントの例: IP アドレス範囲から 1 つのファイルにアクセスする

次の署名付き Cookie 内のカスタムポリシーの例では、UTC の 2023 年 1 月 1 日午前 10 時 00 分まで、範囲 192.0.2.0/24 の IP アドレスから、ユーザーがファイル https://d111111abcdef8.cloudfront.net/game_download.zip にアクセスできることを指定しています。

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/game_download.zip",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.0/24"
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1357034400
                }
            }
        }
    ]
}
```

ポリシーステートメントの例: IP アドレス範囲からディレクトリ内のすべてのファイルにアクセスする

以下のカスタムポリシーの例では、Resource パラメータの * ワイルドカード文字が示すとおり、training ディレクトリ内のあらゆるファイルを対象とする署名付き Cookie を作成できます。UTC の 2013 年 1 月 1 日午前 10 時 00 分まで、範囲 192.0.2.0/24 の IP アドレスから、ユーザーはファイルにアクセスできます。

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/training/*",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.0/24"
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1357034400
                }
            }
        }
    ]
}
```

このポリシーを使用する各署名付き Cookie には、たとえば次のように、特定のファイルを識別するベース URL が含まれます。

https://d111111abcdef8.cloudfront.net/training/orientation.pdf

ポリシーステートメントの例: キーペア ID に関連付けられたすべてのファイルに 1 つの IP アドレスからアクセスする

以下のカスタムポリシーの例では、Resource パラメータの * ワイルドカード文字が示すとおり、あらゆるディストリビューションに関連付けられたあらゆるファイルを対象とする署名付き Cookie を設定できます。ユーザーは IP アドレス 192.0.2.10/32 を使用する必要があります。(CIDR 表記の値 192.0.2.10/32 は 1 つの IP アドレス 192.0.2.10 を参照します)。ファイルは、UTC の 2013 年 1 月 1 日午前 10 時 00 分から UTC の 2013 年 1 月 2 日午前 10 時 00 分まで使用できます。

```
{
    "Statement": [
        {
            "Resource": "https://*",
```

```
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.10/32"
                },
                "DateGreaterThan": {
                    "AWS:EpochTime": 1357034400
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1357120800
                }
            }
        }
    ]
}
```

このポリシーを含める各署名付き Cookie には、たとえば次のように、特定の CloudFront ディストリビューション内の特定のファイルを識別するベース URL が含まれます。

`https://d111111abcdef8.cloudfront.net/training/orientation.pdf`

署名付き Cookie にはキーペア ID も含まれます。キーペア ID は、ベース URL に指定されたディストリビューション (d111111abcdef8.cloudfront.net) 内の信頼されたキーグループに関連付けられる必要があります。

カスタムポリシーを使用する署名付き Cookie の署名を作成する

カスタムポリシーを使用する署名付き Cookie の署名は、ハッシュ化、署名、および base64 エンコードが行われたバージョンのポリシーステートメントです。

ポリシーステートメントのハッシュ化、署名、およびエンコードを行う方法の詳細および例については、以下の各資料を参照してください。

- Linux コマンドおよび OpenSSL を使用した Base64 エンコードおよび暗号化
- 署名付き URL の署名を作成するためのコード例

カスタムポリシーを使用して署名付き Cookie の署名を作成するには

1. 「カスタムポリシーを使用する署名付き URL のポリシーステートメントを作成するには」の手順で作成した JSON ポリシーステートメントを、SHA-1 ハッシュ関数と RSA を使用してハッシュ化し、署名します。空白は含まれていないが、まだ base64 エンコードされていないバージョンのポリシーステートメントを使用します。

ハッシュ関数に必要なプライベートキーには、対応するパブリックキーがディストリビューション内のアクティブな信頼されたキーグループにあるものを使用してください。

> **Note**
>
> ポリシーステートメントをハッシュ化および署名するための方法は、プログラミング言語およびプラットフォームによって異なります。サンプルコードについては、「署名付き URL の署名を作成するためのコード例」を参照してください。

2. ハッシュ化および署名された文字列から、空白 (タブや改行文字を含む) を削除します。
3. MIME base64 エンコーディングを使用して文字列を Base64 エンコードします。詳細については、RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies の Section 6.8, Base64 Content-Transfer-Encoding を参照してください。
4. URL クエリ文字列内の無効な文字を有効な文字で置き換えます。次の表に無効な文字と有効な文字を示します。

無効な文字 (置換元)	有効な文字 (置換先)
+	- (ハイフン)
=	_ (下線)
/	~ (チルダ)

5. 結果の値を、`Set-Cookie` の名前と値のペアの `CloudFront-Signature=` ヘッダーに含めて、「カスタムポリシーを使用して署名付き Cookie を設定するには」に戻り、`Set-Cookie` の `CloudFront-Key-Pair-Id` ヘッダーを追加します。

Linux コマンドおよび OpenSSL を使用した Base64 エンコードおよび暗号化

次の Linux コマンドラインのコマンドおよび OpenSSL を使用して、ポリシーステートメントをハッシュ化して署名します。次に、署名を base64 エンコードし、URL クエリ文字列パラメータでの無効な文字を有効な文字置き換えます。

OpenSSL の詳細については、「https://www.openssl.org」にアクセスしてください。

```
cat policy | tr -d "\n" | tr -d " \t\n\r" | openssl sha1 -sign private_key.pem |
 openssl base64 -A | tr -- '+=/' '-_~'
```

上記のコマンドでは:

- `cat` が `policy` ファイルを読み取ります。
- `tr -d "\n" | tr -d " \t\n\r"` は、`cat` によって追加された空白や改行文字を削除します。
- OpenSSL は、SHA-1 を使用してファイルをハッシュし、RSA とプライベートキーファイル `private_key.pem` を使用して署名します。
- OpenSSL は、ハッシュされ、署名されたポリシーステートメントを base64 でエンコードします。
- `tr` は、URL クエリ文字列パラメータの無効な文字を有効な文字で置き換えます。

署名の作成方法を示すコード例については、「署名付き URL の署名を作成するためのコード例」を参照してください。

署名付き URL の署名を作成するためのコード例

このセクションには、署名付き URL の署名の作成方法を示す、ダウンロード可能なアプリケーションの例が含まれます。例は、Perl、PHP、C#、および Java で使用できます。任意の例を使用して、署名付き URL を作成できます。Perl スクリプトは Linux および macOS プラットフォームで実行されます。PHP の例は、PHP が実行されているあらゆるサーバーで動作します。C# の例では、.NET Framework が使用されます。

JavaScript (Node.js) のコードサンプルについては、AWS デベロッパーブログの「Creating Amazon CloudFront Signed URLs in Node.js」を参照してください。

Python のコードサンプルについては、「AWS SDK for Python (Boto3) API リファレンス」の「Amazon CloudFront の署名付き URL を生成する」と、Boto3 GitHub リポジトリにある「こちらのサンプルコード」を参照してください。

トピック

- Perl を使用して URL 署名を作成する
- PHP を使用して URL 署名を作成する

- C# と .NET Framework を使用して URL 署名を作成する
- Java を使用して URL 署名を作成する

Perl を使用して URL 署名を作成する

このセクションには、プライベートコンテンツの署名を作成するために使用できる Linux/Mac プラットフォームの Perl スクリプトが含まれます。署名を作成するには、コマンドライン引数に CloudFront URL、署名者のプライベートキーへのパス、キー ID、および URL の有効期限切れ日付を指定して、スクリプトを実行します。このツールでは、署名付き URL のデコードを行うこともできます。

> **Note**
>
> URL 署名の作成は、署名付き URL を使用してプライベートコンテンツを供給するためのプロセスの 1 パートにすぎません。エンドツーエンドの処理の詳細については、「署名付き URL を使用する」を参照してください。

トピック

- 署名付き URL を作成する Perl スクリプトのソース

署名付き URL を作成する Perl スクリプトのソース

次の Perl ソースコードを使用して、CloudFront の署名付き URL を作成できます。コード内のコメントに、コマンドラインスイッチおよびこのツールの各種機能の情報が含まれています。

```
#!/usr/bin/perl -w

# Copyright 2008 Amazon Technologies, Inc.  Licensed under the Apache License, Version
 2.0 (the "License");
# you may not use this file except in compliance with the License. You may obtain a
 copy of the License at:
#
# https://aws.amazon.com/apache2.0
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 KIND, either express or implied.
# See the License for the specific language governing permissions and limitations under
 the License.
```

```
=head1 cfsign.pl

cfsign.pl - A tool to generate and verify Amazon CloudFront signed URLs

=head1 SYNOPSIS

This script uses an existing RSA key pair to sign and verify Amazon CloudFront signed
 URLs

View the script source for details as to which CPAN packages are required beforehand.

For help, try:

cfsign.pl --help

URL signing examples:

cfsign.pl --action encode --url https://images.my-website.com/gallery1.zip --policy
 sample_policy.json --private-key privkey.pem --key-pair-id mykey

cfsign.pl --action encode --url https://images.my-website.com/gallery1.zip --expires
 1257439868 --private-key privkey.pem --key-pair-id mykey

URL decode example:

cfsign.pl --action decode --url "http//mydist.cloudfront.net/?Signature=AGO-
PgxkYo99MkJFHvjfGXjG1QDEXeaDb4Qtzmy85wqyJjK7eKojQWa4BCRcow__&Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc29
Pair-Id=mykey"


To generate an RSA key pair, you can use openssl and the following commands:

# Generate a 2048 bit key pair
openssl genrsa -out private-key.pem 2048
openssl rsa -in private-key.pem -pubout -out public-key.pem


=head1 OPTIONS

=over 8

=item B<--help>
```

```
Print a help message and exits.

=item B<--action> [action]

The action to execute.  action can be one of:

  encode - Generate a signed URL (using a canned policy or a user policy)
  decode - Decode a signed URL

=item B<--url>

The URL to en/decode

=item B<--stream>

The stream to en/decode

=item B<--private-key>

The path to your private key.

=item B<--key-pair-id>

The key pair identifier.

=item B<--policy>

The CloudFront policy document.

=item B<--expires>

The Unix epoch time when the URL is to expire. If both this option and
the --policy option are specified, --policy will be used. Otherwise, this
option alone will use a canned policy.

=back

=cut

use strict;
use warnings;

# you might need to use CPAN to get these modules.
# run perl -MCPAN -e "install <module>" to get them.
```

```
# The openssl command line will also need to be in your $PATH.
use File::Temp qw/tempfile/;
use File::Slurp;
use Getopt::Long;
use IPC::Open2;
use MIME::Base64 qw(encode_base64 decode_base64);
use Pod::Usage;
use URI;

my $CANNED_POLICY
    = '{"Statement":[{"Resource":"<RESOURCE>","Condition":{"DateLessThan":
{"AWS:EpochTime":<EXPIRES>}}}]}';

my $POLICY_PARAM      = "Policy";
my $EXPIRES_PARAM     = "Expires";
my $SIGNATURE_PARAM   = "Signature";
my $KEY_PAIR_ID_PARAM = "Key-Pair-Id";

my $verbose = 0;
my $policy_filename = "";
my $expires_epoch = 0;
my $action = "";
my $help = 0;
my $key_pair_id = "";
my $url = "";
my $stream = "";
my $private_key_filename = "";

my $result = GetOptions("action=s"      => \$action,
                        "policy=s"      => \$policy_filename,
                        "expires=i"     => \$expires_epoch,
                        "private-key=s" => \$private_key_filename,
                        "key-pair-id=s" => \$key_pair_id,
                        "verbose"       => \$verbose,
                        "help"          => \$help,
                        "url=s"         => \$url,
                        "stream=s"      => \$stream,
                    );

if ($help or !$result) {
    pod2usage(1);
    exit;
}
```

```
if ($url eq "" and $stream eq "") {
    print STDERR "Must include a stream or a URL to encode or decode with the --stream
 or --url option\n";
    exit;
}

if ($url ne "" and $stream ne "") {
    print STDERR "Only one of --url and --stream may be specified\n";
    exit;
}

if ($url ne "" and !is_url_valid($url)) {
    exit;
}

if ($stream ne "") {
    exit unless is_stream_valid($stream);

    # The signing mechanism is identical, so from here on just pretend we're
    # dealing with a URL
    $url = $stream;
}

if ($action eq "encode") {
    # The encode action will generate a private content URL given a base URL,
    # a policy file (or an expires timestamp) and a key pair id parameter
    my $private_key;
    my $public_key;
    my $public_key_file;

    my $policy;
    if ($policy_filename eq "") {
        if ($expires_epoch == 0) {
            print STDERR "Must include policy filename with --policy argument or an
 expires" .
                         "time using --expires\n";
        }

        $policy = $CANNED_POLICY;
        $policy =~ s/<EXPIRES>/$expires_epoch/g;
        $policy =~ s/<RESOURCE>/$url/g;
    } else {
        if (! -e $policy_filename) {
            print STDERR "Policy file $policy_filename does not exist\n";
```

```
            exit;
        }
        $expires_epoch = 0; # ignore if set
        $policy = read_file($policy_filename);
    }

    if ($private_key_filename eq "") {
        print STDERR "You must specific the path to your private key file with --
private-key\n";
        exit;
    }

    if (! -e $private_key_filename) {
        print STDERR "Private key file $private_key_filename does not exist\n";
        exit;
    }

    if ($key_pair_id eq "") {
        print STDERR "You must specify a key pair id with --key-pair-id\n";
        exit;
    }

    my $encoded_policy = url_safe_base64_encode($policy);
    my $signature = rsa_sha1_sign($policy, $private_key_filename);
    my $encoded_signature = url_safe_base64_encode($signature);

    my $generated_url = create_url($url, $encoded_policy, $encoded_signature,
 $key_pair_id, $expires_epoch);


    if ($stream ne "") {
        print "Encoded stream (for use within a swf):\n" . $generated_url . "\n";
        print "Encoded and escaped stream (for use on a webpage):\n" .
 escape_url_for_webpage($generated_url) . "\n";
    } else {
        print "Encoded URL:\n" . $generated_url . "\n";
    }
} elsif ($action eq "decode") {
    my $decoded = decode_url($url);
    if (!$decoded) {
        print STDERR "Improperly formed URL\n";
        exit;
    }
```

```
    print_decoded_url($decoded);
} else {
    # No action specified, print help.  But only if this is run as a program (caller
 will be empty)
    pod2usage(1) unless caller();
}

# Decode a private content URL into its component parts
sub decode_url {
    my $url = shift;

    if ($url =~ /(.*)\?(.*)/) {
        my $base_url = $1;
        my $params = $2;

        my @unparsed_params = split(/&/, $params);
        my %params = ();
        foreach my $param (@unparsed_params) {
            my ($key, $val) = split(/=/, $param);
            $params{$key} = $val;
        }

        my $encoded_signature = "";
        if (exists $params{$SIGNATURE_PARAM}) {
            $encoded_signature = $params{"Signature"};
        } else {
            print STDERR "Missing Signature URL parameter\n";
            return 0;
        }

        my $encoded_policy = "";
        if (exists $params{$POLICY_PARAM}) {
            $encoded_policy = $params{$POLICY_PARAM};
        } else {
            if (!exists $params{$EXPIRES_PARAM}) {
                print STDERR "Either the Policy or Expires URL parameter needs to be
 specified\n";
                return 0;
            }

            my $expires = $params{$EXPIRES_PARAM};

            my $policy = $CANNED_POLICY;
            $policy =~ s/<EXPIRES>/$expires/g;
```

```
            my $url_without_cf_params = $url;
            $url_without_cf_params =~ s/$SIGNATURE_PARAM=[^&]*&?//g;
            $url_without_cf_params =~ s/$POLICY_PARAM=[^&]*&?//g;
            $url_without_cf_params =~ s/$EXPIRES_PARAM=[^&]*&?//g;
            $url_without_cf_params =~ s/$KEY_PAIR_ID_PARAM=[^&]*&?//g;

            if ($url_without_cf_params =~ /(.*)\?$/) {
                $url_without_cf_params = $1;
            }

            $policy =~ s/<RESOURCE>/$url_without_cf_params/g;

            $encoded_policy = url_safe_base64_encode($policy);
        }

        my $key = "";
        if (exists $params{$KEY_PAIR_ID_PARAM}) {
            $key = $params{$KEY_PAIR_ID_PARAM};
        } else {
            print STDERR "Missing $KEY_PAIR_ID_PARAM parameter\n";
            return 0;
        }

        my $policy = url_safe_base64_decode($encoded_policy);

        my %ret = ();
        $ret{"base_url"} = $base_url;
        $ret{"policy"} = $policy;
        $ret{"key"} = $key;

        return \%ret;
    } else {
        return 0;
    }
}

# Print a decoded URL out
sub print_decoded_url {
    my $decoded = shift;

    print "Base URL: \n" . $decoded->{"base_url"} . "\n";
    print "Policy: \n" . $decoded->{"policy"} . "\n";
    print "Key: \n" . $decoded->{"key"} . "\n";
```

```
}

# Encode a string with base 64 encoding and replace some invalid URL characters
sub url_safe_base64_encode {
    my ($value) = @_;

    my $result = encode_base64($value);
    $result =~ tr|+=/|-_~|;

    return $result;
}

# Decode a string with base 64 encoding.  URL-decode the string first
# followed by reversing any special character ("+=/") translation.
sub url_safe_base64_decode {
    my ($value) = @_;

    $value =~ s/%([0-9A-Fa-f]{2})/chr(hex($1))/eg;
    $value =~ tr|-_~|+=/|;

    my $result = decode_base64($value);

    return $result;
}

# Create a private content URL
sub create_url {
    my ($path, $policy, $signature, $key_pair_id, $expires) = @_;

    my $result;
    my $separator = $path =~ /\?/ ? '&' : '?';
    if ($expires) {
        $result = "$path$separator$EXPIRES_PARAM=$expires&$SIGNATURE_PARAM=$signature&
$KEY_PAIR_ID_PARAM=$key_pair_id";
    } else {
        $result = "$path$separator$POLICY_PARAM=$policy&$SIGNATURE_PARAM=$signature&
$KEY_PAIR_ID_PARAM=$key_pair_id";
    }
    $result =~ s/\n//g;

    return $result;
}

# Sign a document with given private key file.
```

```
# The first argument is the document to sign
# The second argument is the name of the private key file
sub rsa_sha1_sign {
    my ($to_sign, $pvkFile) = @_;
    print "openssl sha1 -sign $pvkFile $to_sign\n";

    return write_to_program($pvkFile, $to_sign);
}

# Helper function to write data to a program
sub write_to_program {
my ($keyfile, $data) = @_;
unlink "temp_policy.dat" if (-e "temp_policy.dat");
unlink "temp_sign.dat" if (-e "temp_sign.dat");

write_file("temp_policy.dat", $data);

system("openssl dgst -sha1 -sign \"$keyfile\" -out temp_sign.dat temp_policy.dat");

my $output = read_file("temp_sign.dat");

    return $output;
}

# Read a file into a string and return the string
sub read_file {
    my ($file) = @_;

    open(INFILE, "<$file") or die("Failed to open $file: $!");
    my $str = join('', <INFILE>);
    close INFILE;

    return $str;
}

sub is_url_valid {
    my ($url) = @_;

    # HTTP distributions start with http[s]:// and are the correct thing to sign
    if ($url =~ /^https?:\/\//) {
        return 1;
    } else {
        print STDERR "CloudFront requires absolute URLs for HTTP distributions\n";
        return 0;
```

```
    }
}

sub is_stream_valid {
    my ($stream) = @_;

    if ($stream =~ /^rtmp:\/\// or $stream =~ /^\/?cfx\/st/) {
        print STDERR "Streaming distributions require that only the stream name is
 signed.\n";
        print STDERR "The stream name is everything after, but not including, cfx/st/
\n";
        return 0;
    } else {
        return 1;
    }
}

# flash requires that the query parameters in the stream name are url
# encoded when passed in through javascript, etc.  This sub handles the minimal
# required url encoding.
sub escape_url_for_webpage {
    my ($url) = @_;

    $url =~ s/\?/%3F/g;
    $url =~ s/=/%3D/g;
    $url =~ s/&/%26/g;

    return $url;
}

1;
```

PHP を使用して URL 署名を作成する

PHP を実行するすべてのウェブサーバーでは、この PHP サンプルコードを使用して、CloudFront のプライベートディストリビューション用のポリシーステートメントと署名を作成できます。この完全なサンプルでは、CloudFront ストリーミングを使用してビデオストリームを再生する署名付き URL リンクが含まれた、実際に動作するウェブページを作成します。完全なサンプルは、https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/samples/demo-php.zip からダウンロードできます。

AWS SDK for PHP の `UrlSigner` クラスを使用して署名付き URL を作成することもできます。詳細については、AWS SDK for PHP API リファレンスの「Class UrlSigner」を参照してください。

> **Note**
>
> URL 署名の作成は、署名付き URL を使用してプライベートコンテンツを供給するためのプロセスの 1 パートにすぎません。プロセス全体の詳細については、「署名付き URL を使用する」を参照してください。

トピック

- 例: RSA SHA-1 署名
- 例: 既定ポリシーを作成する
- 例: カスタムポリシーを作成する
- 完全なサンプルコード

例: RSA SHA-1 署名

以下のサンプルコードでは、関数 `rsa_sha1_sign` によってポリシーステートメントのハッシュと署名を行います。必要な引数は、ポリシーステートメントと、ディストリビューションの信頼されたキーグループにあるパブリックキーに対応するプライベートキーです。次に、`url_safe_base64_encode` 関数で、URL で使用可能なバージョンの署名を作成します。

```
function rsa_sha1_sign($policy, $private_key_filename) {
    $signature = "";

    // load the private key
    $fp = fopen($private_key_filename, "r");
    $priv_key = fread($fp, 8192);
    fclose($fp);
    $pkeyid = openssl_get_privatekey($priv_key);

    // compute signature
    openssl_sign($policy, $signature, $pkeyid);

    // free the key from memory
    openssl_free_key($pkeyid);

    return $signature;
```

```
}

function url_safe_base64_encode($value) {
    $encoded = base64_encode($value);
    // replace unsafe characters +, = and / with
    // the safe characters -, _ and ~
    return str_replace(
        array('+', '=', '/'),
        array('-', '_', '~'),
        $encoded);
}
```

例: 既定ポリシーを作成する

以下のサンプルコードでは、署名用の既定ポリシーステートメントを作成します。既定ポリシーの詳細については、「既定ポリシーを使用して署名付き URL を作成する」を参照してください。

> **Note**
>
> $expires 変数は、文字列ではなく整数である必要がある日時スタンプです。

```
function get_canned_policy_stream_name($video_path, $private_key_filename,
 $key_pair_id, $expires) {
    // this policy is well known by CloudFront, but you still need to sign it,
    // since it contains your parameters
    $canned_policy = '{"Statement":[{"Resource":"' . $video_path . '","Condition":
{"DateLessThan":{"AWS:EpochTime":'. $expires . '}}}]}';

    // sign the canned policy
    $signature = rsa_sha1_sign($canned_policy, $private_key_filename);
    // make the signature safe to be included in a url
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, null, $encoded_signature,
 $key_pair_id, $expires);
    // url-encode the query string characters to work around a flash player bug
    return encode_query_params($stream_name);
}
```

例: カスタムポリシーを作成する

以下のサンプルコードでは、署名用のカスタムポリシーステートメントを作成します。カスタムポリシーの詳細については、「カスタムポリシーを使用して署名付き URL を作成する」を参照してください。

```
function get_custom_policy_stream_name($video_path, $private_key_filename,
 $key_pair_id, $policy) {
    // sign the policy
    $signature = rsa_sha1_sign($policy, $private_key_filename);
    // make the signature safe to be included in a url
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, $encoded_policy, $encoded_signature,
 $key_pair_id, null);
    // url-encode the query string characters to work around a flash player bug
    return encode_query_params($stream_name);
}
```

完全なサンプルコード

次のコードサンプルでは、PHP で CloudFront 署名付き URL を作成する詳細な方法を示します。この完全なサンプルは、https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/samples/demo-php.zip からダウンロードできます。

次の例では、$policy Condition エレメントを変更して、IPv4 アドレス範囲と IPv6 アドレス範囲の両方を許可できます。例については、「Amazon Simple Storage Service ユーザーガイド」の「IAM ポリシーでの IPv6 アドレスの使用」を参照してください。

```
<?php

function rsa_sha1_sign($policy, $private_key_filename) {
    $signature = "";

    // load the private key
    $fp = fopen($private_key_filename, "r");
    $priv_key = fread($fp, 8192);
    fclose($fp);
    $pkeyid = openssl_get_privatekey($priv_key);

    // compute signature
```

```
    openssl_sign($policy, $signature, $pkeyid);

    // free the key from memory
    openssl_free_key($pkeyid);

    return $signature;
}

function url_safe_base64_encode($value) {
    $encoded = base64_encode($value);
    // replace unsafe characters +, = and / with the safe characters -, _ and ~
    return str_replace(
        array('+', '=', '/'),
        array('-', '_', '~'),
        $encoded);
}

function create_stream_name($stream, $policy, $signature, $key_pair_id, $expires) {
    $result = $stream;
    // if the stream already contains query parameters, attach the new query parameters
 to the end
    // otherwise, add the query parameters
    $separator = strpos($stream, '?') == FALSE ? '?' : '&';
    // the presence of an expires time means we're using a canned policy
    if($expires) {
        $result .= $path . $separator . "Expires=" . $expires . "&Signature=" .
 $signature . "&Key-Pair-Id=" . $key_pair_id;
    }
    // not using a canned policy, include the policy itself in the stream name
    else {
        $result .= $path . $separator . "Policy=" . $policy . "&Signature=" .
 $signature . "&Key-Pair-Id=" . $key_pair_id;
    }

    // new lines would break us, so remove them
    return str_replace('\n', '', $result);
}

function encode_query_params($stream_name) {
    // Adobe Flash Player has trouble with query parameters being passed into it,
    // so replace the bad characters with their URL-encoded forms
    return str_replace(
        array('?', '=', '&'),
        array('%3F', '%3D', '%26'),
```

```
        $stream_name);
}

function get_canned_policy_stream_name($video_path, $private_key_filename,
 $key_pair_id, $expires) {
    // this policy is well known by CloudFront, but you still need to sign it, since it
 contains your parameters
    $canned_policy = '{"Statement":[{"Resource":"' . $video_path . '","Condition":
{"DateLessThan":{"AWS:EpochTime":'. $expires . '}}}]}';
    // the policy contains characters that cannot be part of a URL, so we base64 encode
 it
    $encoded_policy = url_safe_base64_encode($canned_policy);
    // sign the original policy, not the encoded version
    $signature = rsa_sha1_sign($canned_policy, $private_key_filename);
    // make the signature safe to be included in a URL
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, null, $encoded_signature,
 $key_pair_id, $expires);
    // URL-encode the query string characters to support Flash Player
    return encode_query_params($stream_name);
}

function get_custom_policy_stream_name($video_path, $private_key_filename,
 $key_pair_id, $policy) {
    // the policy contains characters that cannot be part of a URL, so we base64 encode
 it
    $encoded_policy = url_safe_base64_encode($policy);
    // sign the original policy, not the encoded version
    $signature = rsa_sha1_sign($policy, $private_key_filename);
    // make the signature safe to be included in a URL
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, $encoded_policy, $encoded_signature,
 $key_pair_id, null);
    // URL-encode the query string characters to support Flash Player
    return encode_query_params($stream_name);
}


// Path to your private key.  Be very careful that this file is not accessible
// from the web!
```

```
$private_key_filename = '/home/test/secure/example-priv-key.pem';
$key_pair_id = 'K2JCJMDEHXQW5F';

$video_path = 'example.mp4';

$expires = time() + 300; // 5 min from now
$canned_policy_stream_name = get_canned_policy_stream_name($video_path,
 $private_key_filename, $key_pair_id, $expires);

$client_ip = $_SERVER['REMOTE_ADDR'];
$policy =
'{'.
    '"Statement":['.
        '{'.
            '"Resource":"'. $video_path . '",'.
            '"Condition":{'.
                '"IpAddress":{"AWS:SourceIp":"' . $client_ip . '/32"},'.
                '"DateLessThan":{"AWS:EpochTime":' . $expires . '}'.
            '}'.
        '}'.
    ']' .
    '}';
$custom_policy_stream_name = get_custom_policy_stream_name($video_path,
 $private_key_filename, $key_pair_id, $policy);

?>

<html>

<head>
    <title>CloudFront</title>
<script type='text/javascript' src='https://example.cloudfront.net/player/
swfobject.js'></script>
</head>

<body>
    <h1>Amazon CloudFront</h1>
    <h2>Canned Policy</h2>
    <h3>Expires at <?= gmdate('Y-m-d H:i:s T', $expires) ?></h3>
    <br />

    <div id='canned'>The canned policy video will be here</div>
```

```
    <h2>Custom Policy</h2>
    <h3>Expires at <?= gmdate('Y-m-d H:i:s T', $expires) ?> only viewable by IP <?=
 $client_ip ?></h3>
    <div id='custom'>The custom policy video will be here</div>

    <!-- ************** Have to update the player.swf path to a real JWPlayer instance.
    The fake one means that external people cannot watch the video right now -->
    <script type='text/javascript'>
  var so_canned = new SWFObject('https://files.example.com/
player.swf','mpl','640','360','9');
  so_canned.addParam('allowfullscreen','true');
  so_canned.addParam('allowscriptaccess','always');
  so_canned.addParam('wmode','opaque');
  so_canned.addVariable('file','<?= $canned_policy_stream_name ?>');
  so_canned.addVariable('streamer','rtmp://example.cloudfront.net/cfx/st');
        so_canned.write('canned');

  var so_custom = new SWFObject('https://files.example.com/
player.swf','mpl','640','360','9');
  so_custom.addParam('allowfullscreen','true');
  so_custom.addParam('allowscriptaccess','always');
  so_custom.addParam('wmode','opaque');
  so_custom.addVariable('file','<?= $custom_policy_stream_name ?>');
  so_custom.addVariable('streamer','rtmp://example.cloudfront.net/cfx/st');
  so_custom.write('custom');
    </script>
</body>

</html>
```

次の資料も参照してください:

- Perl を使用して URL 署名を作成する
- C# と .NET Framework を使用して URL 署名を作成する
- Java を使用して URL 署名を作成する

C# と .NET Framework を使用して URL 署名を作成する

このセクションの C# の例では、既定およびカスタムのポリシーステートメントを使用して CloudFront プライベートディストリビューションの署名を作成する方法を示すサンプルアプリケー

ションを実装します。これらの例には、.NET アプリケーションに便利な AWS SDK for .NET に基づくユーティリティ関数が含まれています。

AWS SDK for .NET を使用して署名付き URL と署名付き Cookie を作成することもできます。AWS SDK for .NET API リファレンスで、以下のトピックを参照してください。

- 署名付き URL – AmazonCloudFrontUrlSigner
- 署名付き Cookie – AmazonCloudFrontCookieSigner

コードをダウンロードするには、C# による署名コードにアクセスしてください。

> **Note**
>
> URL 署名の作成は、署名付き URL を使用してプライベートコンテンツを供給するためのプロセスの 1 パートにすぎません。プロセス全体の詳細については、「署名付き URL を使用する」を参照してください。署名付き Cookie の使用の詳細については、「署名付き Cookie を使用する」を参照してください。

.NET Framework で RSA キーを使用する

.NET Framework で RSA キーを使用するには、AWS 提供の .pem ファイルを .NET Framework が使用する XML 形式に変換する必要があります。

変換後、RSA プライベートキーファイルの形式は以下のようになります。

Example : XML .NET Framework 形式の RSA プライベートキー

```
<RSAKeyValue>
  <Modulus>
    wO5IvYCP5UcoCKDo1dcspoMehWBZcyfs9QEzGi6Oe5y+ewGr1oW+vB2GPB
    ANBiVPcUHTFWhwaIBd3oglmF0lGQljP/jOfmXHUK2kUUnLnJp+oOBL2NiuFtqcW6h/L5lIpD8Yq+NRHg
    Ty4zDsyr2880MvXv88yEFURCkqEXAMPLE=
  </Modulus>
  <Exponent>AQAB</Exponent>
  <P>
    5bmKDaTz
    npENGVqz4Cea8XPH+sxt+2VaAwYnsarVUoSBeVt8WLloVuZGG9IZYmH5KteXEu7fZveYd9UEXAMPLE==
  </P>
  <Q>
    1v9l/WN1a1N3rOK4VGoCokx7kR2SyTMSbZgF9IWJNOugR/WZw7HTnjipO3c9dy1Ms9pUKwUF4
```

```
    6d7049EXAMPLE==
  </Q>
  <DP>
    RgrSKuLWXMyBH+/l1Dx/I4tXuAJIrlPyo+VmiOc7b5NzHptkSHEPfR9s1
    OK0VqjknclqCJ3Ig86OMEtEXAMPLE==
  </DP>
  <DQ>
    pjPjvSFw+RoaTu0pgCA/jwW/FGyfN6iim1RFbkT4
    z49DZb2IM885f3vf35eLTaEYRYUHQgZtChNEV0TEXAMPLE==
  </DQ>
  <InverseQ>
    nkvOJTg5QtGNgWb9i
    cVtzrL/1pFEOHbJXwEJdU99N+7sMK+1066DL/HSBUCD63qD4USpnf0myc24in0EXAMPLE==</InverseQ>
  <D>
      Bc7mp7XYHynuPZxChjWNJZIq+A73gm0ASDv6At7F8Vi9r0xUlQe/v0AQS3ycN8QlyR4XMbzMLYk
      3yjxFDXo4ZKQtOGzLGteCU2srANiLv26/imXA8FVidZftTAtLviWQZBVPTeYIA69ATUYPEq0a5u5wjGy
      UOij9OWyuEXAMPLE=
   </D>
</RSAKeyValue>
```

C# における既定ポリシーの署名方法

以下の C# コードは、以下を実行して、既定ポリシーを使用する署名付き URL を作成します。

- ポリシーステートメントを作成する。
- SHA1 を使用してポリシーステートメントをハッシュ化し、RSA とプライベートキー (信頼されたキーグループにあるパブリックキーに対応するもの) を使用して、結果に署名します。
- ハッシュ化および署名されたポリシーステートメントを base64 エンコードし、特殊文字を置き換えて文字列を URL リクエストパラメータとして使用できるようにする。
- 値を連結する。

完全な実装については、C# による署名コードの例を参照してください。

> **Note**
>
> CloudFront にパブリックキーをアップロードすると、keyId が返されます。詳細については、「6 &Key-Pair-Id」を参照してください。

Example : C# における既定ポリシーの署名方法

```
public static string ToUrlSafeBase64String(byte[] bytes)
{
    return System.Convert.ToBase64String(bytes)
        .Replace('+', '-')
        .Replace('=', '_')
        .Replace('/', '~');
}

public static string CreateCannedPrivateURL(string urlString,
    string durationUnits, string durationNumber, string pathToPolicyStmnt,
    string pathToPrivateKey, string keyId)
{
    // args[] 0-thisMethod, 1-resourceUrl, 2-seconds-minutes-hours-days
    // to expiration, 3-numberOfPreviousUnits, 4-pathToPolicyStmnt,
    // 5-pathToPrivateKey, 6-keyId

    TimeSpan timeSpanInterval = GetDuration(durationUnits, durationNumber);

    // Create the policy statement.
    string strPolicy = CreatePolicyStatement(pathToPolicyStmnt,
        urlString,
        DateTime.Now,
        DateTime.Now.Add(timeSpanInterval),
        "0.0.0.0/0");
    if ("Error!" == strPolicy) return "Invalid time frame." +
        "Start time cannot be greater than end time.";

    // Copy the expiration time defined by policy statement.
    string strExpiration = CopyExpirationTimeFromPolicy(strPolicy);

    // Read the policy into a byte buffer.
    byte[] bufferPolicy = Encoding.ASCII.GetBytes(strPolicy);

    // Initialize the SHA1CryptoServiceProvider object and hash the policy data.
    using (SHA1CryptoServiceProvider
        cryptoSHA1 = new SHA1CryptoServiceProvider())
    {
        bufferPolicy = cryptoSHA1.ComputeHash(bufferPolicy);

        // Initialize the RSACryptoServiceProvider object.
        RSACryptoServiceProvider providerRSA = new RSACryptoServiceProvider();
        XmlDocument xmlPrivateKey = new XmlDocument();
```

```
        // Load your private key, which you created by converting your
        // .pem file to the XML format that the .NET framework uses.
        // Several tools are available.
        xmlPrivateKey.Load(pathToPrivateKey);

        // Format the RSACryptoServiceProvider providerRSA and
        // create the signature.
        providerRSA.FromXmlString(xmlPrivateKey.InnerXml);
        RSAPKCS1SignatureFormatter rsaFormatter =
            new RSAPKCS1SignatureFormatter(providerRSA);
        rsaFormatter.SetHashAlgorithm("SHA1");
        byte[] signedPolicyHash = rsaFormatter.CreateSignature(bufferPolicy);

        // Convert the signed policy to URL-safe base64 encoding and
        // replace unsafe characters + = / with the safe characters - _ ~
        string strSignedPolicy = ToUrlSafeBase64String(signedPolicyHash);

        // Concatenate the URL, the timestamp, the signature,
        // and the key pair ID to form the signed URL.
        return urlString +
            "?Expires=" +
            strExpiration +
            "&Signature=" +
            strSignedPolicy +
            "&Key-Pair-Id=" +
            keyId;
    }
}
```

C# におけるカスタムポリシーの署名方法

以下の C# コードは、以下の手順を実行して、カスタムポリシーを使用する署名付き URL を作成します。

1. ポリシーステートメントを作成する。
2. ポリシーステートメントを Base64 エンコードし、特殊文字を置き換えて文字列を URL リクエストパラメータとして使用できるようにする。
3. SHA1 を使用してポリシーステートメントをハッシュ化し、RSA とプライベートキー (信頼されたキーグループにあるパブリックキーに対応するもの) を使用して、結果を暗号化します。
4. ハッシュ化されたポリシーステートメントを base64 エンコードし、特殊文字を置き換えて文字列を URL リクエストパラメータとして使用できるようにする。

5. 値を連結する。

完全な実装については、C# による署名コードの例を参照してください。

> **Note**
>
> CloudFront にパブリックキーをアップロードすると、keyId が返されます。詳細については、「6 &Key-Pair-Id」を参照してください。

Example : C# におけるカスタムポリシーの署名方法

```
public static string ToUrlSafeBase64String(byte[] bytes)
{
    return System.Convert.ToBase64String(bytes)
        .Replace('+', '-')
        .Replace('=', '_')
        .Replace('/', '~');
}

public static string CreateCustomPrivateURL(string urlString,
    string durationUnits, string durationNumber, string startIntervalFromNow,
    string ipaddress, string pathToPolicyStmnt, string pathToPrivateKey,
    string keyId)
{
    // args[] 0-thisMethod, 1-resourceUrl, 2-seconds-minutes-hours-days
    // to expiration, 3-numberOfPreviousUnits, 4-starttimeFromNow,
    // 5-ip_address, 6-pathToPolicyStmt, 7-pathToPrivateKey, 8-keyId

    TimeSpan timeSpanInterval = GetDuration(durationUnits, durationNumber);
    TimeSpan timeSpanToStart = GetDurationByUnits(durationUnits,
        startIntervalFromNow);
    if (null == timeSpanToStart)
        return "Invalid duration units." +
            "Valid options: seconds, minutes, hours, or days";

    string strPolicy = CreatePolicyStatement(
        pathToPolicyStmnt, urlString, DateTime.Now.Add(timeSpanToStart),
        DateTime.Now.Add(timeSpanInterval), ipaddress);
```

```
    // Read the policy into a byte buffer.
    byte[] bufferPolicy = Encoding.ASCII.GetBytes(strPolicy);

    // Convert the policy statement to URL-safe base64 encoding and
    // replace unsafe characters + = / with the safe characters - _ ~

    string urlSafePolicy = ToUrlSafeBase64String(bufferPolicy);

    // Initialize the SHA1CryptoServiceProvider object and hash the policy data.
    byte[] bufferPolicyHash;
    using (SHA1CryptoServiceProvider cryptoSHA1 =
        new SHA1CryptoServiceProvider())
    {
        bufferPolicyHash = cryptoSHA1.ComputeHash(bufferPolicy);

        // Initialize the RSACryptoServiceProvider object.
        RSACryptoServiceProvider providerRSA = new RSACryptoServiceProvider();
        XmlDocument xmlPrivateKey = new XmlDocument();

        // Load your private key, which you created by converting your
        // .pem file to the XML format that the .NET framework uses.
        // Several tools are available.
        xmlPrivateKey.Load(pathToPrivateKey);

        // Format the RSACryptoServiceProvider providerRSA
        // and create the signature.
        providerRSA.FromXmlString(xmlPrivateKey.InnerXml);
        RSAPKCS1SignatureFormatter RSAFormatter =
            new RSAPKCS1SignatureFormatter(providerRSA);
        RSAFormatter.SetHashAlgorithm("SHA1");
        byte[] signedHash = RSAFormatter.CreateSignature(bufferPolicyHash);

        // Convert the signed policy to URL-safe base64 encoding and
        // replace unsafe characters + = / with the safe characters - _ ~
        string strSignedPolicy = ToUrlSafeBase64String(signedHash);

        return urlString +
            "?Policy=" +
            urlSafePolicy +
            "&Signature=" +
            strSignedPolicy +
            "&Key-Pair-Id=" +
            keyId;
```

```
    }
}
```

署名生成のためのユーティリティメソッド

以下のメソッドは、ファイルからポリシーステートメントを取得し、署名生成の期間を解析します。

Example : 署名生成のためのユーティリティメソッド

```
public static string CreatePolicyStatement(string policyStmnt,
   string resourceUrl,
   DateTime startTime,
   DateTime endTime,
   string ipAddress)

{
   // Create the policy statement.
   FileStream streamPolicy = new FileStream(policyStmnt, FileMode.Open,
 FileAccess.Read);
   using (StreamReader reader = new StreamReader(streamPolicy))
   {
      string strPolicy = reader.ReadToEnd();

      TimeSpan startTimeSpanFromNow = (startTime - DateTime.Now);
      TimeSpan endTimeSpanFromNow = (endTime - DateTime.Now);
      TimeSpan intervalStart =
         (DateTime.UtcNow.Add(startTimeSpanFromNow)) -
         new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);
      TimeSpan intervalEnd =
         (DateTime.UtcNow.Add(endTimeSpanFromNow)) -
         new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);

      int startTimestamp = (int)intervalStart.TotalSeconds; // START_TIME
      int endTimestamp = (int)intervalEnd.TotalSeconds;  // END_TIME

      if (startTimestamp > endTimestamp)
         return "Error!";

      // Replace variables in the policy statement.
      strPolicy = strPolicy.Replace("RESOURCE", resourceUrl);
      strPolicy = strPolicy.Replace("START_TIME", startTimestamp.ToString());
      strPolicy = strPolicy.Replace("END_TIME", endTimestamp.ToString());
      strPolicy = strPolicy.Replace("IP_ADDRESS", ipAddress);
      strPolicy = strPolicy.Replace("EXPIRES", endTimestamp.ToString());
```

```
      return strPolicy;
   }
}

public static TimeSpan GetDuration(string units, string numUnits)
{
   TimeSpan timeSpanInterval = new TimeSpan();
   switch (units)
   {
      case "seconds":
         timeSpanInterval = new TimeSpan(0, 0, 0, int.Parse(numUnits));
         break;
      case "minutes":
         timeSpanInterval = new TimeSpan(0, 0, int.Parse(numUnits), 0);
         break;
      case "hours":
         timeSpanInterval = new TimeSpan(0, int.Parse(numUnits), 0 ,0);
         break;
      case "days":
         timeSpanInterval = new TimeSpan(int.Parse(numUnits),0 ,0 ,0);
         break;
      default:
         Console.WriteLine("Invalid time units;" +
            "use seconds, minutes, hours, or days");
         break;
   }
   return timeSpanInterval;
}

private static TimeSpan GetDurationByUnits(string durationUnits,
   string startIntervalFromNow)
{
   switch (durationUnits)
   {
      case "seconds":
         return new TimeSpan(0, 0, int.Parse(startIntervalFromNow));
      case "minutes":
         return new TimeSpan(0, int.Parse(startIntervalFromNow), 0);
      case "hours":
         return new TimeSpan(int.Parse(startIntervalFromNow), 0, 0);
      case "days":
         return new TimeSpan(int.Parse(startIntervalFromNow), 0, 0, 0);
      default:
         return new TimeSpan(0, 0, 0, 0);
```

```
    }
}

public static string CopyExpirationTimeFromPolicy(string policyStatement)
{
    int startExpiration = policyStatement.IndexOf("EpochTime");
    string strExpirationRough = policyStatement.Substring(startExpiration +
        "EpochTime".Length);
    char[] digits = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' };

    List<char> listDigits = new List<char>(digits);
    StringBuilder buildExpiration = new StringBuilder(20);

    foreach (char c in strExpirationRough)
    {
        if (listDigits.Contains(c))
            buildExpiration.Append(c);
    }
    return buildExpiration.ToString();
}
```

以下も参照してください。

- Perl を使用して URL 署名を作成する
- PHP を使用して URL 署名を作成する
- Java を使用して URL 署名を作成する

Java を使用して URL 署名を作成する

次のコード例に加えて、AWS SDK for Java (バージョン 1) の`CloudFrontUrlSigner` ユーティリティクラスを使用して CloudFront の署名付き URL を作成することもできます。

その他の例については、「AWS SDK コードサンプルコードライブラリ」の「AWS SDK を使用して署名付き URL と Cookie を作成する」を参照してください。

> **Note**
>
> 署名付き URL の作成は、CloudFront を使用してプライベートコンテンツを供給するプロセスの一部にすぎません。プロセス全体の詳細については、「署名付き URL を使用する」を参照してください。

以下の例は、CloudFront の署名付き URL の作成方法を示しています。

Example Java のポリシーおよび署名暗号化メソッド

```
package org.example;

import java.time.Instant;
import java.time.temporal.ChronoUnit;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class Main {

    public static void main(String[] args) throws Exception {
        CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilities.create();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        String resourceUrl = "https://a1b2c3d4e5f6g7.cloudfront.net";
        String keyPairId = "K1UA3WV15I7JSD";
        CannedSignerRequest cannedRequest = CannedSignerRequest.builder()
                .resourceUrl(resourceUrl)
                .privateKey(new java.io.File("/path/to/private_key.pem").toPath())
                .keyPairId(keyPairId)
                .expirationDate(expirationDate)
                .build();
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedRequest);
        String url = signedUrl.url();
        System.out.println(url);

    }
}
```

以下も参照してください。

- Perl を使用して URL 署名を作成する
- PHP を使用して URL 署名を作成する
- C# と .NET Framework を使用して URL 署名を作成する

AWS オリジンへのアクセスを制限する

以下の利点が得られるような方法で CloudFront Front および AWS オリジンを設定できます。

- AWS へのアクセスを制限して、パブリックにアクセスできないようにします。
- ビューワー (ユーザー) が、指定した CloudFront ディストリビューションのみを介して AWS オリジンのコンテンツにアクセスできるようにします。つまり、バケットから直接コンテンツにアクセスしたり、意図しない CloudFront ディストリビューションを介してコンテンツにアクセスしたりできないようにします。

そのためには、認証済みリクエストを AWS オリジンに送信するように CloudFront を設定し、CloudFront からの認証済みリクエストへのアクセスのみを許可するように AWS オリジンを設定します。詳細については、互換性のある AWS オリジンのタイプに関する以下のトピックを参照してください。

トピック

- AWS Elemental MediaPackage v2 オリジンへのアクセスの制限
- AWS Elemental MediaStore オリジンへのアクセスを制限する
- AWS Lambda 関数 URL オリジンへのアクセスを制限する
- Amazon Simple Storage Service オリジンへのアクセスを制限する

AWS Elemental MediaPackage v2 オリジンへのアクセスの制限

CloudFront には、MediaPackage v2 オリジンへのアクセスを制限するためのオリジンアクセスコントロール (OAC) が用意されています。

> **Note**
>
> CloudFront OAC は、MediaPackage v2 のみをサポートしています。MediaPackage v1 はサポートされていません。

トピック

- 新しい OAC の作成
- オリジンアクセスコントロールの詳細設定

新しい OAC の作成

CloudFront で新しい OAC を設定するには、以下のトピックに示す手順を実行します。

トピック

前提条件

OAC を作成して設定する前に、MediaPackage v2 オリジンを持つ CloudFront ディストリビューションが必要です。詳細については、「MediaStore コンテナまたは MediaPackage チャネルを使用する」を参照してください。

MediaPackage v2 オリジンへのアクセス許可を OAC に付与する

CloudFront ディストリビューションで OAC を作成または設定する前に、OAC に MediaPackage v2 オリジンへのアクセス許可があることを確認してください。これは、CloudFront ディストリビューションを作成した後で、ディストリビューション設定で MediaPackage v2 オリジンに OAC を追加する前に行います。

MediaPackage v2 オリジンへのアクセス許可を OAC に付与するには、IAM ポリシーを使用して、CloudFront サービスプリンシパル (`cloudfront.amazonaws.com`) にオリジンへのアクセスを許可します。ポリシーの `Condition` 要素は、MediaPackage v2 オリジンを含む CloudFront ディストリビューションを対象とするリクエストに限り、MediaPackage v2 オリジンへのアクセスを CloudFront に許可します。

Example : CloudFront ディストリビューションへの読み取り専用アクセスを許可する IAM ポリシー

次のポリシーは、CloudFront ディストリビューション (*`E1PDK09ESKHJWT`*) に MediaPackage v2 オリジンへのアクセスを許可します。オリジンは、`Resource` 要素に指定された ARN です。

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

```
            "Effect": "Allow",
            "Principal": {"Service": "cloudfront.amazonaws.com"},
            "Action": "mediapackagev2:GetObject",
            "Resource": "arn:aws:mediapackagev2:us-
east-1:123456789012:channelGroup/channel-group-name/channel/channel-name/
originEndpoint/origin_endpoint_name",
            "Condition": {
                "StringEquals": {"AWS:SourceArn":
 "arn:aws:cloudfront::123456789012:distribution/E1PDK09ESKHJWT"}
            }
        }
    ]
}
```

> **Note**
>
> MediaPackage v2 オリジンへのアクセス許可がないディストリビューションを作成した場合は、CloudFront コンソールで [ポリシーをコピー] を選択し、[エンドポイントのアクセス許可を更新] を選択できます。次に、コピーしたアクセス許可をエンドポイントにアタッチできます。詳細については、「AWS Elemental MediaPackage ユーザーガイド」の「エンドポイントポリシーフィールド」を参照してください。

OAC の作成

OAC を作成するには、AWS Management Console、AWS CloudFormation、AWS CLI、または CloudFront API を使用できます。

Console

OAC を作成するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[オリジンアクセス] を選択します。
3. [コントロール設定を作成] を選択します。
4. [新しい OAC を作成] フォームで、次の操作を実行します。

    a. OAC の名前とオプションの説明を入力します。

b. [署名ビヘイビア] は、デフォルト設定の [署名リクエスト (推奨)] のままにすることをお勧めします。詳細については、「the section called “オリジンアクセスコントロールの詳細設定”」を参照してください。

5. [オリジンタイプ] で、[MediaPackage V2] を選択します。
6. [Create] (作成) を選択します。

> **Tip**
>
> OAC を作成したら、名前を書き留めておきます。次の手順では、この操作を行う必要があります。

ディストリビューションの MediaPackage v2 オリジンに OAC を追加するには

1. CloudFront コンソールを https://console.aws.amazon.com/cloudfront/v4/home で開きます。
2. OAC を追加する先の MediaPackage V2 オリジンがあるディストリビューションを選択し、[オリジン] タブを選択します。
3. OAC を追加する先の MediaPackage v2 オリジンを選択し、[編集] を選択します。
4. オリジンの [Protocol] (プロトコル) として [HTTPS only] (HTTPS のみ) を選択します。
5. [オリジンアクセスコントロール] ドロップダウンから、使用する OAC 名を選択します。
6. [Save changes] (変更の保存) をクリックします。

ディストリビューションは、すべての CloudFront エッジロケーションへのデプロイを開始します。エッジロケーションは、新しい設定を受け取ると、MediaPackage v2 オリジンに送信するすべてのリクエストに署名します。

CloudFormation

AWS CloudFormation で OAC を作成するには、`AWS::CloudFront::OriginAccessControl` リソースタイプを使用します。次の例は、OAC を作成するための AWS CloudFormation テンプレート構文を YAML 形式で示しています。

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
      Description: An optional description for the origin access control
      Name: ExampleOAC
```

```
    OriginAccessControlOriginType: mediapackagev2
    SigningBehavior: always
    SigningProtocol: sigv4
```

詳細については、AWS CloudFormation ユーザーガイドの「AWS::CloudFront::OriginAccessControl」を参照してください。

CLI

AWS Command Line Interface (AWS CLI) を使用してオリジンアクセスコントロールを作成するには、aws cloudfront create-origin-access-control コマンドを使用します。コマンドの入力パラメータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

オリジンアクセスコントロール (入力ファイルを含む CLI) を作成するには

1. 次のコマンドを使用して、origin-access-control.yaml という名前のファイルを作成します。このファイルには、create-origin-access-control コマンドのすべての入力パラメータが含まれます。

```
aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input > origin-access-control.yaml
```

2. 先ほど作成した origin-access-control.yaml ファイルを開きます。ファイルを編集して OAC の名前、説明 (オプション) を追加し、SigningBehavior を always に変更します。その後、ファイルを保存します。

   その他の OAC の設定については、「the section called “オリジンアクセスコントロールの詳細設定”」を参照してください。

3. 次のコマンドを使用して、origin-access-control.yaml ファイルの入力パラメータを使用し、オリジンアクセスコントロールを作成します。

```
aws cloudfront create-origin-access-control --cli-input-yaml file://origin-access-control.yaml
```

   コマンド出力の Id 値を書き留めます。これは、CloudFront ディストリビューションで MediaPackage v2 オリジンに OAC を追加するために必要です。

既存のディストリビューションで MediaPackage v2 オリジンに OAC をアタッチするには (CLI で入力ファイルを使用する場合)

1. 以下のコマンドを使用して、OAC を追加する CloudFront ディストリビューションのディストリビューション設定を保存します。ディストリビューションには MediaPackage v2 オリジンが必要です。

```
aws cloudfront get-distribution-config --id <CloudFront distribution ID> --
output yaml > dist-config.yaml
```

2. 先ほど作成した `dist-config.yaml` という名前のファイルを開きます。ファイルを編集し、以下の変更を加えます。

   - `Origins` オブジェクトで、`OriginAccessControlId` という名前のフィールドに OAC の ID を追加します。
   - `OriginAccessIdentity` という名前のフィールドから値を削除します (存在する場合)。
   - `ETag` フィールドの名前を `IfMatch` に変更します。ただし、フィールドの値は変更しないでください。

   完了したら、ファイルを保存します。

3. オリジンアクセスコントロールを使用するようにディストリビューションを更新するには、次のコマンドを使用します。

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-
input-yaml file://dist-config.yaml
```

ディストリビューションは、すべての CloudFront エッジロケーションへのデプロイを開始します。エッジロケーションは、新しい設定を受け取ると、MediaPackage v2 オリジンに送信するすべてのリクエストに署名します。

API

CloudFront API で OAC を作成するには、CreateOriginAccessControl を使用します。この API コールで指定するフィールドの詳細については、AWS SDK やその他の API クライアントの API リファレンスドキュメントを参照してください。

OAC を作成したら、以下の API コールのいずれかを使用して、OAC をディストリビューションの MediaPackage v2 オリジンにアタッチできます。

- 既存のディストリビューションにアタッチするには、UpdateDistribution を使用します。
- 新しいディストリビューションにアタッチするには、CreateDistribution を使用します。

これらの API コールの両方について、`OriginAccessControlId` フィールドのオリジン内に OAC ID を指定します。これらの API コールで指定するその他フィールドの詳細については、「ディストリビューション設定リファレンス」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

オリジンアクセスコントロールの詳細設定

CloudFront OAC 機能には、特定のユースケースのみを対象とした詳細設定が含まれています。詳細設定が特に必要でない限り、推奨設定を使用してください。

OAC には、[署名ビヘイビア] (コンソール) または `SigningBehavior` (API、CLI、AWS CloudFormation) という名前の設定が含まれています。この設定では、次のオプションを使用できます。

オリジンリクエストに常に署名する (推奨設定)

コンソールの [Sign requests (recommended)] (署名リクエスト (推奨))、または API、CLI、および AWS CloudFormation の `always` という設定を使用することをお勧めします。この設定の場合、CloudFront は MediaPackage v2 オリジンに送信するすべてのリクエストに常に署名します。

オリジンリクエストに署名しない

この設定は、コンソールでは [リクエストに署名しない]、または API、CLI、およびAWS CloudFormation では `never` です。この設定を使用して、この OAC を使用するすべてのディストリビューションですべてのオリジンの OAC をオフにします。OAC を使用するすべてのオリジンとディストリビューションから OAC を 1 つずつ削除する場合と比べて、この設定で時間と労力を節約できます。この設定の場合、CloudFront は MediaPackage v2 オリジンに送信するいずれのリクエストにも署名しません。

> **Warning**
>
> この設定を使用するには、MediaPackage v2 オリジンがパブリックにアクセス可能である必要があります。この設定を、パブリックにアクセスできない MediaPackage v2 オリジンで使用すると、CloudFront はオリジンにアクセスできません。MediaPackage v2 オリジンは CloudFront にエラーを返し、CloudFront はこれらのエラーをビューワーに渡します。詳細については、「AWS Elemental MediaPackage ユーザーガイド」の「MediaPackage のポリシーとアクセス許可」で MediaPackage v2 ポリシーの例を参照してください。

ビューワー (クライアント) の **Authorization** ヘッダーを上書きしない

この設定は、コンソールでは [認証ヘッダーを上書きしない] で、API、CLI、および AWS CloudFormation では `no-override` です。この設定は、対応するビューワーリクエストに `Authorization` ヘッダーに含まれていない場合にのみ、CloudFront がオリジンリクエストに署名するよう指定する場合に使用します。この設定では、ビューワーリクエストが存在するが、ビューワーリクエストに `Authorization` ヘッダーが含まれていないときにオリジンリクエストに署名する (独自の `Authorization` ヘッダーを追加) ときに、CloudFront はビューワーリクエストから `Authorization` ヘッダーを渡します。

> **Warning**
>
> ビューワーリクエストから `Authorization` ヘッダーを渡すには、このオリジンアクセスコントロールに関連付けられた MediaPackage v2 オリジンを使用するすべてのキャッシュビヘイビアで、`Authorization` ヘッダーをキャッシュポリシーに追加する必要があります。

AWS Elemental MediaStore オリジンへのアクセスを制限する

CloudFront には、AWS Elemental MediaStore オリジンへのアクセスを制限するためのオリジンアクセスコントロール (OAC) が用意されています。

トピック

- 新しいオリジンアクセスコントロールを作成する
- オリジンアクセスコントロールの詳細設定

新しいオリジンアクセスコントロールを作成する

次のトピックで説明されているステップを実行して、CloudFront で新しいオリジンアクセスコントロールを設定します。

トピック

前提条件

オリジンアクセスコントロール (OAC) を作成して設定する前に、MediaStore オリジンを持つ CloudFront ディストリビューションが必要です。

MediaStore へのアクセス許可をオリジンアクセスコントロールに付与する

CloudFront ディストリビューションでオリジンアクセスコントロール (OAC) を作成または設定する前に、OAC に MediaStore オリジンへのアクセス許可があることを確認してください。これは CloudFront ディストリビューションを作成した後、ディストリビューション設定で MediaStore オリジンに OAC を追加する前に行います。

MediaStore オリジンへのアクセス許可を OAC に付与するには、MediaStore コンテナポリシーを使用して、CloudFront サービスプリンシパル (`cloudfront.amazonaws.com`) に、オリジンへのアクセスを許可します。ポリシーの `Condition` 要素を使用して、MediaStore オリジンを含む CloudFront ディストリビューションに代わってリクエストが行われた場合にのみ MediaStore コンテナへのアクセスを CloudFront に許可します。

以下の例は、MediaStore オリジンへのアクセスを CloudFront OAC に許可する MediaStore コンテナポリシーを示しています。

Example CloudFront OAC への読み取り専用アクセスを許可する MediaStore コンテナポリシー

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "AllowCloudFrontServicePrincipalReadOnly",
                "Effect": "Allow",
                "Principal": {
```

```
                  "Service": "cloudfront.amazonaws.com"
                },
                "Action": [
                  "mediastore:GetObject"
                ],
                "Resource":
 "arn:aws:mediastore:<region>:111122223333:container/<container name>/*",
                "Condition": {
                    "StringEquals": {
                      "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
                    },
                    "Bool": {
                      "aws:SecureTransport": "true"
                    }
                }
            }
        ]
}
```

Example CloudFront OAC への読み取りおよび書き込みアクセスを許可する MediaStore コンテナポリシー

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "AllowCloudFrontServicePrincipalReadWrite",
                "Effect": "Allow",
                "Principal": {
                  "Service": "cloudfront.amazonaws.com"
                },
                "Action": [
                  "mediastore:GetObject",
                  "mediastore:PutObject"
                ],
                "Resource":
 "arn:aws:mediastore:<region>:111122223333:container/<container name>/*",
                "Condition": {
                    "StringEquals": {
                      "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
                    },
```

```
                    "Bool": {
                      "aws:SecureTransport": "true"
                    }
                }
            }
        ]
}
```

> **Note**
>
> 書き込みアクセスを許可するには、許可されている HTTP メソッドの設定により、CloudFront ディストリビューションの動作設定に PUT を含める必要があります。

オリジンアクセスコントロールを作成する

OAC を作成するには、AWS Management Console、AWS CloudFormation、AWS CLI、または CloudFront API を使用できます。

Console

オリジンアクセスコントロールを作成するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[オリジンアクセス] を選択します。
3. [コントロール設定を作成] を選択します。
4. [コントロール設定を作成] フォームで、以下の操作を行います。
   a. [詳細] ペインに、[名前] と (オプションで) オリジンアクセスコントロールの [説明] を入力します。
   b. [設定] ペインでは、デフォルト設定である [Sign requests (recommended)] (署名リクエスト (推奨)) をそのまま使用することをお勧めします。詳細については、「the section called “オリジンアクセスコントロールの詳細設定”」を参照してください。
5. [Origin type] (オリジンタイプ) ドロップダウンから [MediaStore] を選択します。
6. [Create] (作成) を選択します。

   OAC を作成したら、[名前] を書き留めておきます。次の手順では、この操作を行う必要があります。

ディストリビューションの MediaStore オリジンにオリジンアクセスコントロールを追加するには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。
2. OAC を追加する先の MediaStore オリジンがあるディストリビューションを選択し、[Origins] (オリジン) タブを選択します。
3. OAC を追加する先の MediaStore オリジンを選択し、[Edit] (編集) を選択します。
4. オリジンの [Protocol] (プロトコル) として [HTTPS only] (HTTPS のみ) を選択します。
5. [Origin access control] (オリジンアクセスコントロール) ドロップダウンメニューから、使用する OAC を選択します。
6. [Save changes] (変更の保存) をクリックします。

ディストリビューションは、すべての CloudFront エッジロケーションへのデプロイを開始します。エッジロケーションは、新しい設定を受け取ると、MediaStore バケットオリジンに送信するすべてのリクエストに署名します。

CloudFormation

AWS CloudFormation を使用してオリジンアクセスコントロール (OAC) を作成するには、`AWS::CloudFront::OriginAccessControl` リソースタイプを使用します。以下の例は、オリジンアクセスコントロールを作成するための AWS CloudFormation テンプレート構文を YAML 形式で示しています。

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
      Description: An optional description for the origin access control
      Name: ExampleOAC
      OriginAccessControlOriginType: mediastore
      SigningBehavior: always
      SigningProtocol: sigv4
```

詳細については、AWS CloudFormation ユーザーガイドの「AWS::CloudFront::OriginAccessControl」を参照してください。

CLI

AWS Command Line Interface (AWS CLI) を使用してオリジンアクセスコントロールを作成するには、aws cloudfront create-origin-access-control コマンドを使用します。コマンドの入力パラ

メータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

オリジンアクセスコントロール (入力ファイルを含む CLI) を作成するには

1. 次のコマンドを使用して、origin-access-control.yaml という名前のファイルを作成します。このファイルには、create-origin-access-control コマンドのすべての入力パラメータが含まれます。

```
aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
 origin-access-control.yaml
```

2. 先ほど作成した origin-access-control.yaml ファイルを開きます。ファイルを編集して OAC の名前、説明 (オプション) を追加し、SigningBehavior を always に変更します。その後、ファイルを保存します。

   その他の OAC の設定については、「the section called “オリジンアクセスコントロールの詳細設定”」を参照してください。

3. 次のコマンドを使用して、origin-access-control.yaml ファイルの入力パラメータを使用し、オリジンアクセスコントロールを作成します。

```
aws cloudfront create-origin-access-control --cli-input-yaml file://origin-
access-control.yaml
```

   コマンド出力の Id 値を書き留めます。これは、CloudFront ディストリビューションの MediaStore オリジンに OAC を追加するために必要です。

OAC を既存のディストリビューションの MediaStore オリジンにアタッチするには (CLI で入力ファイルを使用する場合)

1. 以下のコマンドを使用して、OAC を追加する CloudFront ディストリビューションのディストリビューション設定を保存します。ディストリビューションには MediaStoore オリジンが必要です。

```
aws cloudfront get-distribution-config --id <CloudFront distribution ID> --
output yaml > dist-config.yaml
```

2. 先ほど作成した dist-config.yaml という名前のファイルを開きます。ファイルを編集し、以下の変更を加えます。

   - Origins オブジェクトで、OriginAccessControlId という名前のフィールドに OAC の ID を追加します。
   - OriginAccessIdentity という名前のフィールドから値を削除します (存在する場合)。
   - ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。

   完了したら、ファイルを保存します。

3. オリジンアクセスコントロールを使用するようにディストリビューションを更新するには、次のコマンドを使用します。

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-
input-yaml file://dist-config.yaml
```

ディストリビューションは、すべての CloudFront エッジロケーションへのデプロイを開始します。エッジロケーションは、新しい設定を受け取ると、MediaStore オリジンに送信するすべてのリクエストに署名します。

API

CloudFront API でオリジンアクセスコントロールを作成するには、CreateOriginAccessControl を使用します。この API コールで指定するフィールドの詳細については、AWS SDK やその他の API クライアントの API リファレンスドキュメントを参照してください。

オリジンアクセスコントロールを作成したら、以下の API コールのいずれかを使用して、それをディストリビューションの MediaStore オリジンにアタッチできます。

- 既存のディストリビューションにアタッチするには、UpdateDistribution を使用します。
- 新しいディストリビューションにアタッチするには、CreateDistribution を使用します。

これらの API コールの両方について、OriginAccessControlId フィールドのオリジン内にオリジンアクセスコントロール ID を指定します。これらの API コールで指定するその他フィール

ドの詳細については、「ディストリビューション設定リファレンス」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

オリジンアクセスコントロールの詳細設定

CloudFront オリジンアクセスコントロール機能には、特定のユースケースのみを対象とした詳細設定が含まれています。詳細設定が特に必要でない限り、推奨設定を使用してください。

オリジンアクセスコントロールには、[署名動作] (コンソール)、または `SigningBehavior` (API、CLI、および AWS CloudFormation) という設定が含まれています この設定では、次のオプションを使用できます。

オリジンリクエストに常に署名する (推奨設定)

コンソールの [Sign requests (recommended)] (署名リクエスト (推奨))、または API、CLI、および AWS CloudFormation の `always` という設定を使用することをお勧めします。この設定の場合、CloudFront は MediaStore オリジンに送信するすべてのリクエストに常に署名します。

オリジンリクエストに署名しない

この設定は、コンソールでは [リクエストに署名しない]、または API、CLI、およびAWS CloudFormation では `never` です。この設定を使用して、このオリジンアクセスコントロールを使用するすべてのディストリビューションのすべてのオリジンに対して、オリジンアクセスコントロールをオフにします。これにより、オリジンアクセスコントロールを使用するすべてのオリジンとディストリビューションからコントロールを 1 つずつ削除する場合に比べて、時間と労力を節約できます。この設定の場合、CloudFront は MediaStore オリジンに送信するいずれのリクエストにも署名しません。

> **Warning**
>
> この設定を使用するには、MediaStore オリジンがパブリックにアクセス可能である必要があります。パブリックにアクセスできない MediaStore オリジンでこの設定を使用すると、CloudFront はオリジンにアクセスできません。MediaStore オリジンは CloudFront にエラーを返し、CloudFront はこれらのエラーをビューワーに渡します。詳細については、コンテナポリシーの例: HTTPS 経由のパブリック読み取りアクセスを参照してください。

ビューワー (クライアント) の **Authorization** ヘッダーを上書きしない

この設定は、コンソールでは [認証ヘッダーを上書きしない] で、API、CLI、および AWS CloudFormation では no-override です。この設定は、対応するビューワーリクエストに Authorization ヘッダーに含まれていない場合にのみ、CloudFront がオリジンリクエストに署名するよう指定する場合に使用します。この設定では、ビューワーリクエストが存在するが、ビューワーリクエストに Authorization ヘッダーが含まれていないときにオリジンリクエストに署名する (独自の Authorization ヘッダーを追加) ときに、CloudFront はビューワーリクエストから Authorization ヘッダーを渡します。

> **Warning**
>
> ビューワーリクエストから Authorization ヘッダーを渡すには、このオリジンアクセスコントロールに関連付けられた MediaStore オリジンを使用するすべてのキャッシュ動作で、Authorization ヘッダーをキャッシュポリシーに追加する必要があります。

AWS Lambda 関数 URL オリジンへのアクセスを制限する

CloudFront には、Lambda 関数 URL オリジンへのアクセスを制限するためのオリジンアクセスコントロール (OAC) が用意されています。

トピック

- 新しい OAC を作成する
- オリジンアクセスコントロールの詳細設定

新しい OAC を作成する

CloudFront で新しい OAC を設定するには、以下のトピックに示す手順を実行します。

> **Note**
>
> Lambda 関数 URL で PUT メソッドまたは POST メソッドを使用する場合、ユーザーは CloudFront にリクエストを送信するときに x-amz-content-sha256 ヘッダーにペイロードハッシュ値を含める必要があります。Lambda は、署名されていないペイロードをサポートしていません。

トピック

前提条件

OAC を作成して設定する前に、Lambda 関数 URL をオリジンとして持つ CloudFront ディストリビューションが必要です。詳細については、「Lambda 関数 URL を使用する」を参照してください。

Lambda 関数 URL へのアクセス許可を OAC に付与する

CloudFront ディストリビューションで OAC を作成または設定する前に、Lambda 関数 URL へのアクセス許可が OAC にあることを確認します。これは、CloudFront ディストリビューションを作成した後で、ディストリビューション設定で Lambda 関数 URL に OAC を追加する前に行います。

> **Note**
>
> Lambda 関数 URL の IAM ポリシーを更新するには、AWS Command Line Interface (AWS CLI) を使用する必要があります。現時点では、Lambda コンソールでの IAM ポリシーの編集はサポートされていません。

次の AWS CLI コマンドは、CloudFront サービスプリンシパル (`cloudfront.amazonaws.com`) に Lambda 関数 URL へのアクセスを許可します。ポリシーの `Condition` 要素は、Lambda 関数 URL を含む CloudFront ディストリビューションを対象とするリクエストに限り、Lambda へのアクセス許可を CloudFront に付与します。

Example : CloudFront OAC への読み取り専用アクセスを許可するようにポリシーを更新する AWS CLI コマンド

次の AWS CLI コマンドは、CloudFront ディストリビューション (*`E1PDK09ESKHJWT`*) に Lambda *`FUNCTION_URL_NAME`* へのアクセスを許可します。

```
aws lambda add-permission \
--statement-id "AllowCloudFrontServicePrincipal" \
--action "lambda:InvokeFunctionUrl" \
--principal "cloudfront.amazonaws.com" \
```

```
--source-arn "arn:aws:cloudfront::123456789012:distribution/E1PDK09ESKHJWT" \
--function-name FUNCTION_URL_NAME
```

> **Note**
>
> 作成したディストリビューションに Lambda 関数 URL へのアクセス許可がない場合は、CloudFront コンソールで [CLI コマンドをコピー] を選択し、このコマンドをコマンドラインターミナルから入力できます。詳細については、「AWS Lambda デベロッパーガイド」の「AWS のサービスへのアクセス権を関数に付与する」を参照してください。

OAC を作成する

OAC を作成するには、AWS Management Console、AWS CloudFormation、AWS CLI、または CloudFront API を使用できます。

Console

OAC を作成するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[オリジンアクセス] を選択します。
3. [コントロール設定を作成] を選択します。
4. [新しい OAC を作成] フォームで、次の操作を実行します。
   a. OAC の名前とオプションの説明を入力します。
   b. [署名ビヘイビア] は、デフォルト設定の [署名リクエスト (推奨)] のままにすることをお勧めします。詳細については、「the section called “オリジンアクセスコントロールの詳細設定”」を参照してください。
5. [オリジンタイプ] で、[Lambda] を選択します。
6. [Create] (作成) を選択します。

> **Tip**
>
> OAC を作成したら、名前を書き留めておきます。次の手順では、この操作を行う必要があります。

ディストリビューションの Lambda 関数 URL にオリジンアクセスコントロールを追加するには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。
2. OAC を追加する先の Lambda 関数 URL があるディストリビューションを選択し、[オリジン] タブを選択します。
3. OAC を追加する先の Lambda 関数 URL を選択し、[編集] を選択します。
4. オリジンの [Protocol] (プロトコル) として [HTTPS only] (HTTPS のみ) を選択します。
5. [オリジンアクセスコントロール] ドロップダウンから、使用する OAC 名を選択します。
6. [Save changes] (変更の保存) をクリックします。

ディストリビューションは、すべての CloudFront エッジロケーションへのデプロイを開始します。エッジロケーションは、新しい設定を受け取ると、Lambda 関数 URL に送信するすべてのリクエストに署名します。

CloudFormation

AWS CloudFormation で OAC を作成するには、`AWS::CloudFront::OriginAccessControl` リソースタイプを使用します。次の例は、OAC を作成するための AWS CloudFormation テンプレート構文を YAML 形式で示しています。

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
      Description: An optional description for the origin access control
      Name: ExampleOAC
      OriginAccessControlOriginType: lambda
      SigningBehavior: always
      SigningProtocol: sigv4
```

詳細については、AWS CloudFormation ユーザーガイドの「AWS::CloudFront::OriginAccessControl」を参照してください。

CLI

AWS Command Line Interface (AWS CLI) を使用してオリジンアクセスコントロールを作成するには、aws cloudfront create-origin-access-control コマンドを使用します。コマンドの入力パラメータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

オリジンアクセスコントロール (入力ファイルを含む CLI) を作成するには

1. 次のコマンドを使用して、origin-access-control.yaml という名前のファイルを作成します。このファイルには、create-origin-access-control コマンドのすべての入力パラメータが含まれます。

```
aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
 origin-access-control.yaml
```

2. 先ほど作成した origin-access-control.yaml ファイルを開きます。ファイルを編集して OAC の名前、説明 (オプション) を追加し、SigningBehavior を always に変更します。その後、ファイルを保存します。

   その他の OAC の設定については、「the section called “オリジンアクセスコントロールの詳細設定”」を参照してください。

3. 次のコマンドを使用して、origin-access-control.yaml ファイルの入力パラメータを使用し、オリジンアクセスコントロールを作成します。

```
aws cloudfront create-origin-access-control --cli-input-yaml file://origin-
access-control.yaml
```

   コマンド出力の Id 値を書き留めます。これは、CloudFront ディストリビューションの Lambda 関数 URL に OAC を追加するために必要です。

既存のディストリビューションの Lambda 関数 URL に OAC をアタッチするには (CLI で入力ファイルを使用する場合)

1. 以下のコマンドを使用して、OAC を追加する CloudFront ディストリビューションのディストリビューション設定を保存します。ディストリビューションには、オリジンとして Lambda 関数 URL が必要です。

```
aws cloudfront get-distribution-config --id <CloudFront distribution ID> --
output yaml > dist-config.yaml
```

2. 先ほど作成した dist-config.yaml という名前のファイルを開きます。ファイルを編集し、以下の変更を加えます。

- `Origins` オブジェクトで、`OriginAccessControlId` という名前のフィールドに OAC の ID を追加します。
- `OriginAccessIdentity` という名前のフィールドから値を削除します (存在する場合)。
- `ETag` フィールドの名前を `IfMatch` に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. オリジンアクセスコントロールを使用するようにディストリビューションを更新するには、次のコマンドを使用します。

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-
input-yaml file://dist-config.yaml
```

ディストリビューションは、すべての CloudFront エッジロケーションへのデプロイを開始します。エッジロケーションは、新しい設定を受け取ると、Lambda 関数 URL に送信するすべてのリクエストに署名します。

API

CloudFront API で OAC を作成するには、CreateOriginAccessControl を使用します。この API コールで指定するフィールドの詳細については、AWS SDK やその他の API クライアントの API リファレンスドキュメントを参照してください。

OAC を作成したら、以下の API コールのいずれかを使用して、ディストリビューションの Lambda 関数 URL に OAC をアタッチできます。

- 既存のディストリビューションにアタッチするには、UpdateDistribution を使用します。
- 新しいディストリビューションにアタッチするには、CreateDistribution を使用します。

これらの API コールの両方について、`OriginAccessControlId` フィールドのオリジン内に OAC ID を指定します。これらの API コールで指定する他のフィールドの詳細については、AWS SDK または他の API クライアントの API リファレンスドキュメントを参照してください。

オリジンアクセスコントロールの詳細設定

CloudFront OAC 機能には、特定のユースケースのみを対象とした詳細設定が含まれています。詳細設定が特に必要でない限り、推奨設定を使用してください。

OAC には、[署名ビヘイビア] (コンソール) または `SigningBehavior` (API、CLI、AWS CloudFormation) という名前の設定が含まれています。この設定では、次のオプションを使用できます。

オリジンリクエストに常に署名する (推奨設定)

コンソールの [Sign requests (recommended)] (署名リクエスト (推奨))、または API、CLI、および AWS CloudFormation の `always` という設定を使用することをお勧めします。この設定の場合、CloudFront は Lambda 関数 URL に送信するすべてのリクエストに常に署名します。

オリジンリクエストに署名しない

この設定は、コンソールでは [リクエストに署名しない]、または API、CLI、およびAWS CloudFormation では `never` です。この設定を使用して、この OAC を使用するすべてのディストリビューションですべてのオリジンの OAC をオフにします。OAC を使用するすべてのオリジンとディストリビューションから OAC を 1 つずつ削除する場合と比べて、この設定で時間と労力を節約できます。この設定の場合、CloudFront は Lambda 関数 URL に送信するいずれのリクエストにも署名しません。

> **Warning**
>
> この設定を使用するには、Lambda 関数 URL がパブリックにアクセス可能である必要があります。この設定を、パブリックにアクセスできない Lambda 関数 URL で使用すると、CloudFront はオリジンにアクセスできません。Lambda 関数 URL は、CloudFront にエラーを返し、CloudFront はこれらのエラーをビューワーに渡します。詳細については、「AWS Lambda ユーザーガイド」の「Lambda 関数 URL におけるセキュリティと認証モデル」を参照してください。

ビューワー (クライアント) の **`Authorization`** ヘッダーを上書きしない

この設定は、コンソールでは [認証ヘッダーを上書きしない] で、API、CLI、および AWS CloudFormation では `no-override` です。この設定は、対応するビューワーリクエストに `Authorization` ヘッダーに含まれていない場合にのみ、CloudFront がオリジンリクエストに

署名するよう指定する場合に使用します。この設定では、ビューワーリクエストが存在するが、ビューワーリクエストに Authorization ヘッダーが含まれていないときにオリジンリクエストに署名する (独自の Authorization ヘッダーを追加) ときに、CloudFront はビューワーリクエストから Authorization ヘッダーを渡します。

> **Warning**
>
> ビューワーリクエストから Authorization ヘッダーを渡すには、このオリジンアクセスコントロールに関連付けられた Lambda 関数 URL を使用するすべてのキャッシュビヘイビアで、Authorization ヘッダーをキャッシュポリシーに追加する必要があります。

Amazon Simple Storage Service オリジンへのアクセスを制限する

CloudFront には、認証済みリクエストを Amazon S3 オリジンに送信するために、オリジンアクセスコントロール (OAC) とオリジンアクセスアイデンティティ (OAI) という 2 つの方法が用意されています。OAC は、Amazon S3 などのオリジンを保護するのに役立ちます。OAC は以下をサポートしているため、OAC の使用をお勧めします。

- すべての AWS リージョンのすべての Amazon S3 バケット (2022 年 12 月以降に開始されたオプトインリージョンを含む)
- AWS KMS による Amazon S3 サーバー側の暗号化 (SSE-KMS)
- Amazon S3 に対する動的なリクエスト (PUT と DELETE)

オリジンアクセスアイデンティティ (OAI) は、上記のリストのシナリオでは機能しないか、追加の回避策が必要です。以下のトピックでは、Amazon S3 オリジンでオリジンアクセスコントロール (OAC) を使用する方法について説明します。オリジンアクセスアイデンティティ (OAI) からオリジンアクセスコントロール (OAC) への移行方法については、「the section called “オリジンアクセスアイデンティティ (OAI) からオリジンアクセスコントロール (OAC) への移行”」を参照してください。

> **メモ**
>
> - Amazon S3 バケットオリジンで CloudFront OAC を使用する場合は、[Amazon S3 オブジェクト所有権] を、新しい Amazon S3 バケットのデフォルトである [バケット所有者の強制] に設定する必要があります。ACL が必要な場合は、[希望するバケット所有者] を使用し、CloudFront 経由でアップロードされたオブジェクトの制御を維持します。

- オリジンが、ウェブサイトエンドポイントとして設定されている Amazon S3 バケットである場合、CloudFront でカスタムオリジンとして設定する必要があります。つまり、OAC (または OAI) を使用することはできません。OAC は、Lambda@Edge を使用したオリジンリダイレクトをサポートしていません。

トピック

- the section called “新しいオリジンアクセスコントロールを作成する”
- the section called “S3 バケットに OAC がアタッチされたディストリビューションを削除する”
- the section called “オリジンアクセスアイデンティティ (OAI) からオリジンアクセスコントロール (OAC) への移行”
- the section called “オリジンアクセスコントロールの詳細設定”

新しいオリジンアクセスコントロールを作成する

次のトピックで説明されているステップを実行して、CloudFront で新しいオリジンアクセスコントロールを設定します。

トピック

- 前提条件
- S3 バケットへのアクセス許可をオリジンアクセスコントロールに付与する
- オリジンアクセスコントロールを作成する

前提条件

オリジンアクセスコントロール (OAC) を作成して設定する前に、Amazon S3 バケットオリジンを持つ CloudFront ディストリビューションが必要です。このオリジンは、ウェブサイトエンドポイントとして設定されたバケットではなく、通常の S3 バケットである必要があります。S3 バケットオリジンを使用した CloudFront ディストリビューションのセットアップの詳細については、「the section called “基本的なディストリビューションの開始方法”」を参照してください。

> **Note**
>
> OAC を使用して S3 バケットオリジンを保護する場合、CloudFront と Amazon S3 間の通信は、特定の設定に関係なく、常に HTTPS を介して行われます。

S3 バケットへのアクセス許可をオリジンアクセスコントロールに付与する

オリジンアクセスコントロール (OAC) を作成したり、CloudFront ディストリビューションで設定したりする前に、OAC に S3 バケットオリジンへのアクセス許可があることを確認してください。これは CloudFront ディストリビューションを作成した後、ディストリビューション設定の S3 オリジンに OAC を追加する前に行います。

S3 バケットへのアクセス許可を OAC に付与するには、S3 バケットポリシーを使用して、CloudFront サービスプリンシパル (`cloudfront.amazonaws.com`) に、バケットへのアクセスを許可します。ポリシーの `Condition` 要素を使用して、S3 オリジンを含む CloudFront ディストリビューションに代わってリクエストが行われた場合にのみ CloudFront がバケットにアクセスできるようにします。

バケットポリシーの追加または変更の詳細については、Amazon S3 ユーザーガイドの「Amazon S3 コンソールを使用したバケットポリシーの追加」を参照してください。

以下の例は、S3 オリジンへのアクセスを CloudFront OAC に許可する S3 バケットポリシーを示しています。

Example CloudFront OAC への読み取り専用アクセスを許可する S3 バケットポリシー

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "AllowCloudFrontServicePrincipalReadOnly",
        "Effect": "Allow",
        "Principal": {
            "Service": "cloudfront.amazonaws.com"
        },
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::<S3 bucket name>/*",
        "Condition": {
            "StringEquals": {
                "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
            }
        }
    }
}
```

Example CloudFront OAC への読み取りと書き込みアクセスを許可する S3 バケットポリシー

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "AllowCloudFrontServicePrincipalReadWrite",
        "Effect": "Allow",
        "Principal": {
            "Service": "cloudfront.amazonaws.com"
        },
        "Action": [
            "s3:GetObject",
            "s3:PutObject"
        ],
        "Resource": "arn:aws:s3:::<S3 bucket name>/*",
        "Condition": {
            "StringEquals": {
                "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
            }
        }
    }
}
```

SSE-KMS

S3 バケットオリジン内のオブジェクトが AWS Key Management Service (SSE-KMS) でのサーバー側の暗号化を使用して暗号化されている場合は、OAC に AWS KMS キーを使用するアクセス許可があることを確認する必要があります。KMS キーを使用するアクセス許可を OAC に付与するには、KMS キーポリシーにステートメントを追加します。キーポリシーを変更する方法については、AWS Key Management Service デベロッパーガイドの「キーポリシーの変更」を参照してください。

次の例は、OAC による KMS キーの使用を許可する KMS キーポリシーステートメントを示しています。

Example CloudFront OAC による SSE-KMS の KMS キーへのアクセスを許可する KMS キーポリシーステートメント

```
{
    "Sid": "AllowCloudFrontServicePrincipalSSE-KMS",
    "Effect": "Allow",
```

```
    "Principal": {
        "Service": [
            "cloudfront.amazonaws.com"
        ]
     },
    "Action": [
        "kms:Decrypt",
        "kms:Encrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*",
    "Condition": {
            "StringEquals": {
                "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
            }
        }
}
```

オリジンアクセスコントロールを作成する

オリジンアクセスコントロール (OAC) を作成するには、AWS Management Console、AWS CloudFormation、AWS CLI、または CloudFront API を使用します。

Console

オリジンアクセスコントロールを作成するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[オリジンアクセス] を選択します。
3. [コントロール設定を作成] を選択します。
4. [コントロール設定を作成] フォームで、以下の操作を行います。
   a. [詳細] ペインに、[名前] と (オプションで) オリジンアクセスコントロールの [説明] を入力します。
   b. [設定] ペインでは、デフォルト設定である [Sign requests (recommended)] (署名リクエスト (推奨)) をそのまま使用することをお勧めします。詳細については、「the section called “オリジンアクセスコントロールの詳細設定”」を参照してください。
5. [Origin type] (オリジンタイプ) ドロップダウンから [S3] を選択します。

6. [Create] (作成) を選択します。

   OAC を作成したら、[名前] を書き留めておきます。次の手順では、この操作を行う必要があります。

ディストリビューションの S3 オリジンにオリジンアクセスコントロールを追加するには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。
2. OAC を追加する S3 オリジンのあるディストリビューションを選択し、[オリジン] タブを選択します。
3. OAC を追加する S3 オリジンを選択し、[編集] を選択します。
4. [オリジンアクセス] で、[オリジンアクセスコントロール設定 (推奨)] を選択します。
5. [Origin access control] (オリジンアクセスコントロール) ドロップダウンメニューから、使用する OAC を選択します。
6. [Save changes] (変更の保存) をクリックします。

ディストリビューションは、すべての CloudFront エッジロケーションへのデプロイを開始します。エッジロケーションは、新しい設定を受け取ると、S3 バケットオリジンに送信するすべてのリクエストに署名します。

CloudFormation

AWS CloudFormation を使用してオリジンアクセスコントロール (OAC) を作成するには、`AWS::CloudFront::OriginAccessControl` リソースタイプを使用します。以下の例は、オリジンアクセスコントロールを作成するための AWS CloudFormation テンプレート構文を YAML 形式で示しています。

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
      Description: An optional description for the origin access control
      Name: ExampleOAC
      OriginAccessControlOriginType: s3
      SigningBehavior: always
      SigningProtocol: sigv4
```

詳細については、AWS CloudFormation ユーザーガイドの「AWS::CloudFront::OriginAccessControl」を参照してください。

CLI

AWS Command Line Interface (AWS CLI) を使用してオリジンアクセスコントロールを作成するには、aws cloudfront create-origin-access-control コマンドを使用します。コマンドの入力パラメータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

オリジンアクセスコントロール (入力ファイルを含む CLI) を作成するには

1. 次のコマンドを使用して、`origin-access-control.yaml` という名前のファイルを作成します。このファイルには、create-origin-access-control コマンドのすべての入力パラメータが含まれます。

```
aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
 origin-access-control.yaml
```

2. 先ほど作成した `origin-access-control.yaml` ファイルを開きます。ファイルを編集して OAC の名前、説明 (オプション) を追加し、`SigningBehavior` を `always` に変更します。その後、ファイルを保存します。

   その他の OAC の設定については、「the section called “オリジンアクセスコントロールの詳細設定”」を参照してください。

3. 次のコマンドを使用して、`origin-access-control.yaml` ファイルの入力パラメータを使用し、オリジンアクセスコントロールを作成します。

```
aws cloudfront create-origin-access-control --cli-input-yaml file://origin-
access-control.yaml
```

   コマンド出力の `Id` 値を書き留めます。これは、CloudFront ディストリビューションの S3 バケットオリジンに OAC を追加するために必要です。

OAC を既存のディストリビューションの S3 バケットオリジンにアタッチするには (入力ファイルを含む CLI)

1. 以下のコマンドを使用して、OAC を追加する CloudFront ディストリビューションのディストリビューション設定を保存します。ディストリビューションには S3 バケットオリジンが必要です。

```
aws cloudfront get-distribution-config --id <CloudFront distribution ID> --
output yaml > dist-config.yaml
```

2. 先ほど作成した dist-config.yaml という名前のファイルを開きます。ファイルを編集し、以下の変更を加えます。

   - Origins オブジェクトで、OriginAccessControlId という名前のフィールドに OAC の ID を追加します。
   - OriginAccessIdentity という名前のフィールドから値を削除します (存在する場合)。
   - ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。

   完了したら、ファイルを保存します。

3. オリジンアクセスコントロールを使用するようにディストリビューションを更新するには、次のコマンドを使用します。

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-
input-yaml file://dist-config.yaml
```

ディストリビューションは、すべての CloudFront エッジロケーションへのデプロイを開始します。エッジロケーションは新しい設定を受け取ると、S3 バケットオリジンに送信するすべてのリクエストに署名します。

API

CloudFront API でオリジンアクセスコントロールを作成するには、CreateOriginAccessControl を使用します。この API コールで指定するフィールドの詳細については、AWS SDK やその他の API クライアントの API リファレンスドキュメントを参照してください。

オリジンアクセスコントロールを作成したら、次の API コールのいずれかを使用して、それをディストリビューションの S3 バケットオリジンにアタッチできます。

- 既存のディストリビューションにアタッチするには、UpdateDistribution を使用します。
- 新しいディストリビューションにアタッチするには、CreateDistribution を使用します。

これらの API コールの両方について、OriginAccessControlId フィールドのオリジン内にオリジンアクセスコントロール ID を指定します。これらの API コールで指定するその他フィールドの詳細については、「ディストリビューション設定リファレンス」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

S3 バケットに OAC がアタッチされたディストリビューションを削除する

S3 バケットに OAC がアタッチされたディストリビューションを削除する必要がある場合は、S3 バケットオリジンを削除する前に、ディストリビューションを削除する必要があります。または、オリジンドメイン名にリージョンを含めます。これが不可能な場合は、削除前にパブリックに切り替えることで、OAC をディストリビューションから削除できます。詳細については、「ディストリビューションを削除する」を参照してください。

オリジンアクセスアイデンティティ (OAI) からオリジンアクセスコントロール (OAC) への移行

レガシーのオリジンアクセスアイデンティティ (OAI) からオリジンアクセスコントロール (OAC) に移行するには、まず S3 バケットオリジンを更新して、OAI と OAC の両方がバケットのコンテンツにアクセスできるようにします。これにより、移行中に CloudFront がバケットにアクセスできなくなる状況を回避できます。OAI と OAC の両方が S3 バケットにアクセスできるようにするには、バケットポリシーを更新し、プリンシパルの種類ごとに 1 つずつ、2 つのステートメントを含めます。

次の S3 バケットポリシー例では、OAI と OAC の両方に S3 オリジンへのアクセスを許可しています。

Example OAI および OAC への読み取り専用アクセスを許可する S3 バケットポリシー

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipalReadOnly",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<S3 bucket name>/*",
            "Condition": {
                "StringEquals": {
```

```
                    "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
                }
            }
        },
        {
            "Sid": "AllowLegacyOAIReadOnly",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity <origin access identity ID>"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<S3 bucket name>/*"
        }
    ]
}
```

S3 オリジンのバケットポリシーを更新して OAI と OAC の両方へのアクセスを許可したら、OAI の代わりに OAC を使用するようにディストリビューション設定を更新できます。詳細については、「the section called “新しいオリジンアクセスコントロールを作成する”」を参照してください。

ディストリビューションが完全にデプロイされたら、OAI へのアクセスを許可するバケットポリシー内のステートメントを削除できます。詳しくは、「the section called “S3 バケットへのアクセス許可をオリジンアクセスコントロールに付与する”」を参照してください。

オリジンアクセスコントロールの詳細設定

CloudFront オリジンアクセスコントロール機能には、特定のユースケースのみを対象とした詳細設定が含まれています。詳細設定が特に必要でない限り、推奨設定を使用してください。

オリジンアクセスコントロールには、[署名動作] (コンソール)、または `SigningBehavior` (API、CLI、および AWS CloudFormation) という設定が含まれています この設定では、次のオプションを使用できます。

オリジンリクエストに常に署名する (推奨設定)

> コンソールの [Sign requests (recommended)] (署名リクエスト (推奨))、または API、CLI、および AWS CloudFormation の `always` という設定を使用することをお勧めします。この設定では、CloudFront は S3 バケットオリジンに送信するすべてのリクエストに常に署名します。

オリジンリクエストに署名しない

この設定は、コンソールでは [リクエストに署名しない]、または API、CLI、およびAWS CloudFormation では `never` です。この設定を使用して、このオリジンアクセスコントロールを使用するすべてのディストリビューションのすべてのオリジンに対して、オリジンアクセスコントロールをオフにします。これにより、オリジンアクセスコントロールを使用するすべてのオリジンとディストリビューションからコントロールを 1 つずつ削除する場合に比べて、時間と労力を節約できます。この設定では、CloudFront は S3 バケットオリジンに送信するリクエストには署名しません。

> **Warning**
>
> この設定を使用するには、S3 バケットオリジンがパブリックにアクセス可能である必要があります。パブリックにアクセスできない S3 バケットオリジンでこの設定を使用すると、CloudFront はオリジンにアクセスできません。S3 バケットオリジンは CloudFront にエラーを返し、CloudFront はそれらのエラーをビューワーに渡します。

ビューワー (クライアント) の **`Authorization`** ヘッダーを上書きしない

この設定は、コンソールでは [認証ヘッダーを上書きしない] で、API、CLI、および AWS CloudFormation では `no-override` です。この設定は、対応するビューワーリクエストに `Authorization` ヘッダーに含まれていない場合にのみ、CloudFront がオリジンリクエストに署名するよう指定する場合に使用します。この設定では、ビューワーリクエストが存在するが、ビューワーリクエストに `Authorization` ヘッダーが含まれていないときにオリジンリクエストに署名する (独自の `Authorization` ヘッダーを追加) ときに、CloudFront はビューワーリクエストから `Authorization` ヘッダーを渡します。

> **Warning**
>
> ビューワーリクエストから `Authorization` ヘッダーを渡すには、このオリジンアクセスコントロールに関連付けられた S3 バケットオリジンを使用するすべてのキャッシュ動作に対して、`Authorization` ヘッダーをキャッシュポリシーに必ず追加する必要があります。

オリジンアクセスアイデンティティを使用する (レガシー、非推奨)

オリジンアクセスアイデンティティの概要

CloudFront のオリジンアクセスアイデンティティ (OAI) はオリジンアクセスコントロール (OAC) と類似した機能を提供しますが、すべてのシナリオで機能するわけではありません。これが、代わりに OAC の使用をお勧めする理由です。具体的には、OAI は以下をサポートしていません。

- オプトインリージョンを含む、すべての AWS リージョンの Amazon S3 バケット
- AWS KMS による Amazon S3 サーバー側の暗号化 (SSE-KMS)
- Amazon S3 に対する動的なリクエスト (PUT、POST、または DELETE)
- 2022 年 12 月以降に開始された新しい AWS リージョン

OAI から OAC への移行に関する詳細については、「the section called “オリジンアクセスアイデンティティ (OAI) からオリジンアクセスコントロール (OAC) への移行”」を参照してください。

Amazon S3 バケット内のファイルを読み取るアクセス許可をオリジンアクセスアイデンティティに付与する

CloudFront コンソールを使用して OAI を作成するか、OAI をディストリビューションに追加する場合、バケットにアクセスするための OAI の許可を与えるように Amazon S3 バケットポリシーを自動的に更新できます。あるいは、このバケットポリシーの手動での作成または更新を選択することができます。どちらの方法を使用する場合でも、アクセス許可を確認して次のことを確認する必要があります。

- CloudFront OAI は、CloudFront 経由でリクエストしているビューワーに代わってバケット内のファイルにアクセスできます。
- ビューワーは、Amazon S3 URL を使用して CloudFront の外のファイルにアクセスすることができません。

> **Important**
>
> CloudFront でサポートされるすべての HTTP メソッドを受け入れて転送するように CloudFront を設定する場合は、必ず CloudFront OAI に目的のアクセス許可を付与してください。例えば、DELETE メソッドを使用するリクエストを受け入れて転送するように CloudFront を設定する場合、適切なファイルのみをビューワーが削除できるように、DELETE リクエストを適切に処理するバケットポリシーを設定してください。

Amazon S3 バケットポリシーを使用する

Amazon S3 バケット内のファイルへのアクセス許可を CloudFront OAI に付与するには、次の方法でバケットポリシーを作成または更新します。

- Amazon S3 コンソールで、Amazon S3 バケットの [アクセス許可] タブを使用する。
- Amazon S3 API の PutBucketPolicy を使用する。
- CloudFront コンソールを使用する。CloudFront コンソールでオリジン設定に OAI を追加するときに、[Yes, update the bucket policy] (はい、バケットポリシーを更新します) を選択すると、ユーザーに代わってバケットポリシーを更新するように CloudFront に指示できます。

バケットポリシーを手動で更新する場合は、次の点を確認してください。

- ポリシーで正しい OAI を `Principal` として指定します。
- ビューワーに代わってオブジェクトにアクセスするために必要なアクセス許可を OAI に付与します。

詳細については、次のセクションを参照してください。

バケットポリシーで OAI を **`Principal`** として指定

Amazon S3 バケットポリシーで `Principal` として OAI を指定するには、OAI の ID を含む OAI の Amazon リソースネーム (ARN) を使用します。例:

```
"Principal": {
    "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity <origin
 access identity ID>"
}
```

CloudFront コンソールの [セキュリティ]、[オリジンアクセス]、[アイデンティティ (レガシー)] で OAI ID を検索します。または、CloudFront API の ListCloudFrontOriginAccessIdentities を使用します。

OAI にアクセス許可を付与

Amazon S3 バケット内のオブジェクトにアクセスするためのアクセス許可を OAI に付与するには、特定の Amazon S3 API オペレーションに関連するポリシーでアクションを使用します。例えば、`s3:GetObject` アクションは、OAI がバケット内のオブジェクトを読み取ることを許可しま

す。詳細については、次のセクションの例を参照するか、Amazon Simple Storage Service ユーザーガイドの「Amazon S3 のアクション」を参照してください。

Amazon S3 バケットポリシーの例

以下の例は、S3 バケットへのアクセスを CloudFront OAI に許可する Amazon S3 バケットポリシーを示しています。

CloudFront コンソールの [セキュリティ]、[オリジンアクセス]、[アイデンティティ (レガシー)] で OAI ID を検索します。または、CloudFront API の ListCloudFrontOriginAccessIdentities を使用します。

Example OAI に読み取りアクセスを許可する Amazon S3 バケットポリシー

次の例では、指定されたバケット (s3:GetObject) 内のオブジェクトの読み取りを OAI に許可します。

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity <origin access identity ID>"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<S3 bucket name>/*"
        }
    ]
}
```

Example OAI に読み取りおよび書き込みアクセスを許可する Amazon S3 バケットポリシー

次の例では、指定されたバケット (s3:GetObject と s3:PutObject) 内のオブジェクトの読み取りおよび書き込みを OAI に許可します。これにより、ビューワーは CloudFront を介して Amazon S3 バケットにファイルをアップロードできます。

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity <origin access identity ID>"
            },
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::<S3 bucket name>/*"
        }
    ]
}
```

Amazon S3 オブジェクト ACL を使用する (非推奨)

> **Important**
>
> Amazon S3 バケットポリシーを使用して OAI へのアクセスを S3 バケットに与えることを推奨します。このセクションで説明したようにアクセスコントロールリスト (ACL) を使用できますが、お勧めしません。
> Amazon S3 では所有権が強化されたバケットに S3 オブジェクトの所有権を設定することを推奨しており、すなわち、バケットとその中のオブジェクトについて ACL が無効になっているということです。この設定を [オブジェクトの所有権] に適用する場合は、バケットポリシーを使用して OAI へのアクセスを与える必要があります (前のセクションを参照)。
> 次のセクションは、ACL を必要とするレガシーユースケースのみを対象としています。

Amazon S3 バケット内のファイルへのアクセス許可を CloudFront OAI に付与するには、次の方法でファイルの ACL を作成または更新します。

- Amazon S3 コンソールで、Amazon S3 オブジェクトの [アクセス許可] タブを使用する。
- Amazon S3 API の PutObjectAcl を使用する。

ACL を使用して OAI へのアクセスを許可する場合、Amazon S3 の正規ユーザー ID を使用して OAI を指定する必要があります。CloudFront コンソールでは、この ID を [セキュリティ]、[オリジンアクセス]、[アイデンティティ (レガシー)] で検索できます。CloudFront API を使用している場合

は、OAI の作成時に返された S3CanonicalUserId 要素の値を使用するか、CloudFront API の ListCloudFrontOriginAccessIdentities を呼び出します。

署名バージョン 4 の認証のみをサポートする Amazon S3 リージョンでオリジンアクセスアイデンティティを使用する

新しい Amazon S3 リージョンでは、リクエストの認証用に署名バージョン 4 を使用する必要があります。(各 Amazon S3 リージョンでサポートされている署名バージョンについては、「AWS 全般のリファレンス」の「Amazon Simple Storage Service エンドポイントおよびクォータ」を参照してください)。オリジンアクセスアイデンティティを使用しており、バケットが、署名バージョン 4 が必要なリージョンの 1 つにある場合、以下の点に注意してください。

- DELETE、GET、HEAD、OPTIONS、および PATCH リクエストは条件なしでサポートされます。
- POST リクエストはサポートされません。

Application Load Balancer へのアクセスを制限する

インターネットに接続している Elastic Load Balancing の Application Load Balancer によって提供されるウェブアプリケーションやその他のコンテンツについて、CloudFront はオブジェクトをキャッシュしてユーザー (閲覧者) に直接提供し、Application Load Balancer の負荷を軽減できます。インターネットに接続しているロードバランサーは、パブリックに解決可能な DNS 名を持ち、クライアントからのリクエストをインターネット経由でターゲットにルーティングします。

CloudFront は、レイテンシーを削減し、分散型サービス拒否 (DDoS) 攻撃を吸収することにも役立ちます。

ただし、ユーザーが CloudFront をバイパスして、Application Load Balancer に直接アクセスできる場合には、これらのメリットを得られません。ですが、 Amazon CloudFront と Application Load Balancer を設定して、ユーザーがApplication Load Balancer に直接アクセスできないようにすることができます。これにより、ユーザーは CloudFront 経由でしか Application Load Balancer にアクセスできず、CloudFront を使用するメリットを得ることができます。

ユーザーが Application Load Balancer に直接アクセスできないようにし、CloudFront 経由だけでアクセスを許可するためには、以下の高レベルのステップを実行します。

1. CloudFront を設定して、Application Load Balancer に送信するリクエストにカスタム HTTP ヘッダーを追加します。
2. カスタム HTTP ヘッダーを含むリクエストだけを転送するように、Application Load Balancer を設定します。

3. （オプション）このソリューションのセキュリティを向上させるためには、HTTPS が必要です。

詳細については、以下のトピックを参照してください。これらのステップを完了すると、ユーザーは CloudFront 経由でしかApplication Load Balancer にアクセスできなくなります。

トピック

- リクエストにカスタム HTTP ヘッダーを追加するように CloudFront を設定する
- 特定のヘッダーを含むリクエストだけを転送するように Application Load Balancer を設定する
- （オプション）このソリューションのセキュリティ向上
- (オプション) CloudFront の AWS マネージドプレフィックスリストを使用してオリジンへのアクセスを制限します。

リクエストにカスタム HTTP ヘッダーを追加するように CloudFront を設定する

オリジン（この場合、Application Load Balancer）に送信するリクエストにカスタム HTTP ヘッダーを追加するように CloudFront を設定できます。

> **Important**
>
> このユースケースは、カスタムヘッダー名と値の機密性維持を信頼しています。ヘッダー名と値が機密でない場合、他の HTTP クライアントは、Application Load Balancer に直接送信するリクエストにヘッダー名や値を含める可能性があります。これにより、リクエストをしていない時に、リクエストが CloudFront から送信されたかのように Application Load Balancer を動作させる可能性があります。これを防ぐためには、カスタムヘッダー名と値を機密にしておきます。

CloudFront コンソール、AWS CloudFormation、または CloudFront API を使用して、オリジンリクエストにカスタム HTTP ヘッダーを追加するように CloudFront を設定できます。

カスタム HTTP ヘッダーの追加 (CloudFront コンソール)

CloudFront コンソールで、オリジン設定のオリジンカスタムヘッダー設定を使用します。次の例に示すように、ヘッダー名とその値を入力します。

> **Note**
>
> この例のヘッダー名と値は、デモンストレーションのためだけに使用されます。製作では、ランダムに生成された値を使用します。ヘッダー名と値は、ユーザー名やパスワードなどの安全な資格情報として扱います。

既存の CloudFront ディストリビューションのオリジンを作成または編集するとき、新しいディストリビューションを作成するときには、オリジンカスタムヘッダー設定を編集できます 。詳細については、ディストリビューションを更新するおよびディストリビューションを作成するを参照してください。

カスタム HTTP ヘッダー (AWS CloudFormation) の追加

AWS CloudFormation テンプレートで、次の例に示すように、`OriginCustomHeaders` プロパティを使用します。

> **Note**
>
> この例のヘッダー名と値は、デモンストレーションのためだけに使用されます。製作では、ランダムに生成された値を使用します。ヘッダー名と値は、ユーザー名やパスワードなどの安全な資格情報として扱います。

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  TestDistribution:
    Type: 'AWS::CloudFront::Distribution'
    Properties:
      DistributionConfig:
        Origins:
          - DomainName: app-load-balancer.example.com
            Id: Example-ALB
            CustomOriginConfig:
              OriginProtocolPolicy: https-only
              OriginSSLProtocols:
```

```
            - TLSv1.2
        OriginCustomHeaders:
           - HeaderName: X-Custom-Header
             HeaderValue: random-value-1234567890
    Enabled: 'true'
    DefaultCacheBehavior:
      TargetOriginId: Example-ALB
      ViewerProtocolPolicy: allow-all
      CachePolicyId: 658327ea-f89d-4fab-a63d-7e88639e58f6
    PriceClass: PriceClass_All
    ViewerCertificate:
      CloudFrontDefaultCertificate: 'true'
```

詳細については、「AWS CloudFormation ユーザーガイド」の [Origin] プロパティと [OriginCustomHeader] プロパティを参照してください。

カスタム HTTP ヘッダーの追加 (CloudFront API)

CloudFront API で、`CustomHeaders` 内部オブジェクト`Origin`を使用します 。詳細については、「Amazon CloudFront API リファレンス」の [CreateDistribution] と [UpdateDistribution]、さらに SDK や他の API クライアントのドキュメントを参照してください。

オリジンカスタムヘッダーとして指定できないヘッダー名がいくつかあります。詳細については、「CloudFront でオリジンリクエストに追加できないカスタムヘッダー」を参照してください。

特定のヘッダーを含むリクエストだけを転送するように Application Load Balancer を設定する

CloudFront を設定して、Application Load Balancer に送信するリクエストにカスタム HTTP ヘッダーを追加した後 (前のセクションを参照) 、このカスタムヘッダーを含むリクエストだけを転送するようにロードバランサーを設定できます。これを行うためには、新しいルールを追加し、ロードバランサーのリスナーでデフォルトルールを変更します。

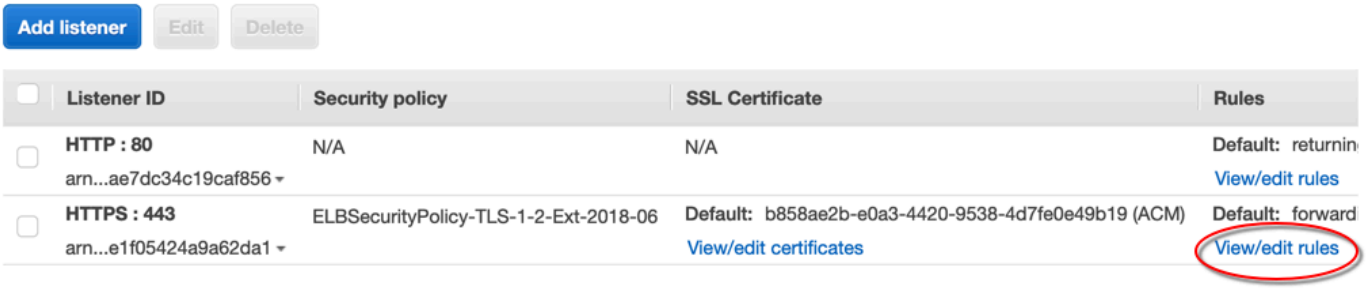
前提条件

次の手順を使用するためには、最低 1 つのリスナーがある Application Load Balancer が必要です。まだ作成していない場合には、Application Load Balancer のユーザーガイドで「Application Load Balancer の作成」を参照してください。

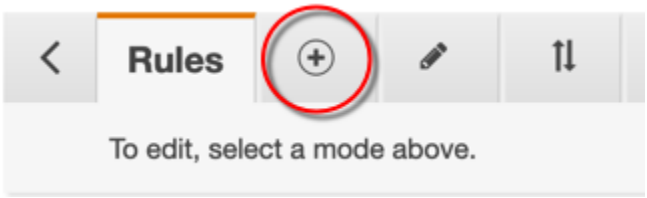
次の手順では、HTTPS リスナーを変更します。同じプロセスを使用して HTTP リスナーを変更できます。

Application Load Balancer リスナーのルールの更新

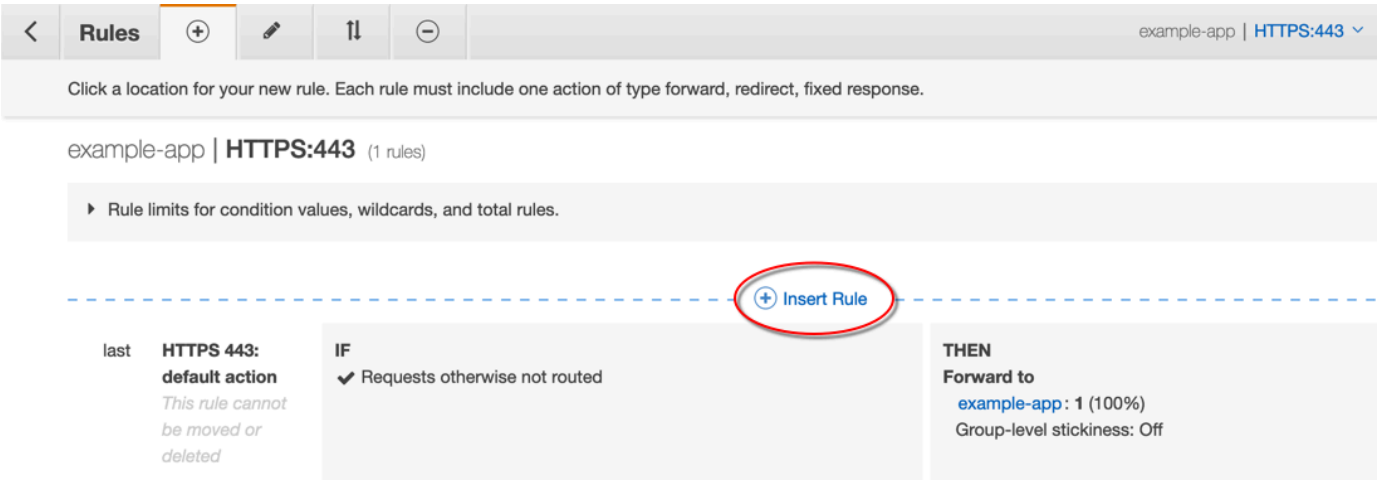
1. Amazon EC2 コンソールで [ロードバランサー] ページを開きます 。
2. CloudFront ディストリビューションのオリジンであるロードバランサーを選択してから、[リスナー] タブを選択します。
3. 変更しているリスナーに、[ルールの表示/編集] を選択します。



4. アイコンを選択してルールを追加します。



5. [Insert Rule] を選択します。

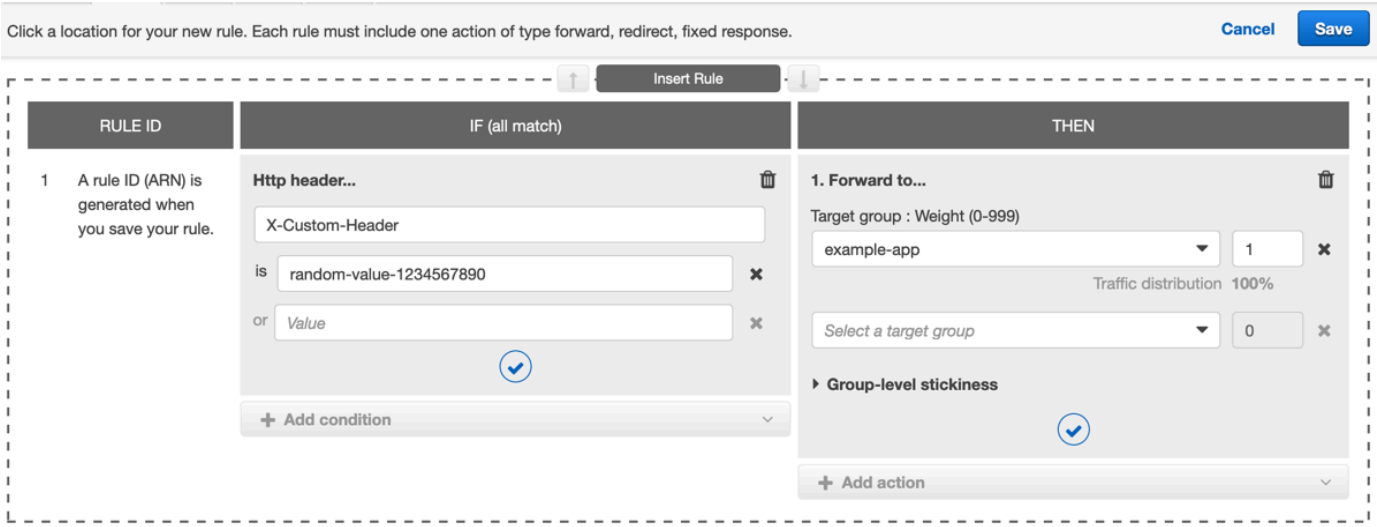


6. 新しいルールでは、次の操作を実行します。

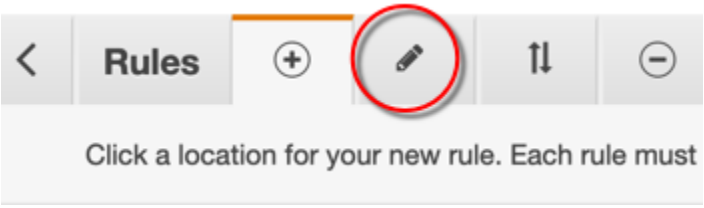
   a. [条件の追加] を選択してから、[Http ヘッダー] を選択します。CloudFront でオリジンカスタムヘッダーとして追加した HTTP ヘッダー名と値を指定します。

   b. [アクションの追加] を選択してから、[転送先] を選択します。リクエストを転送するターゲットグループを選択します。

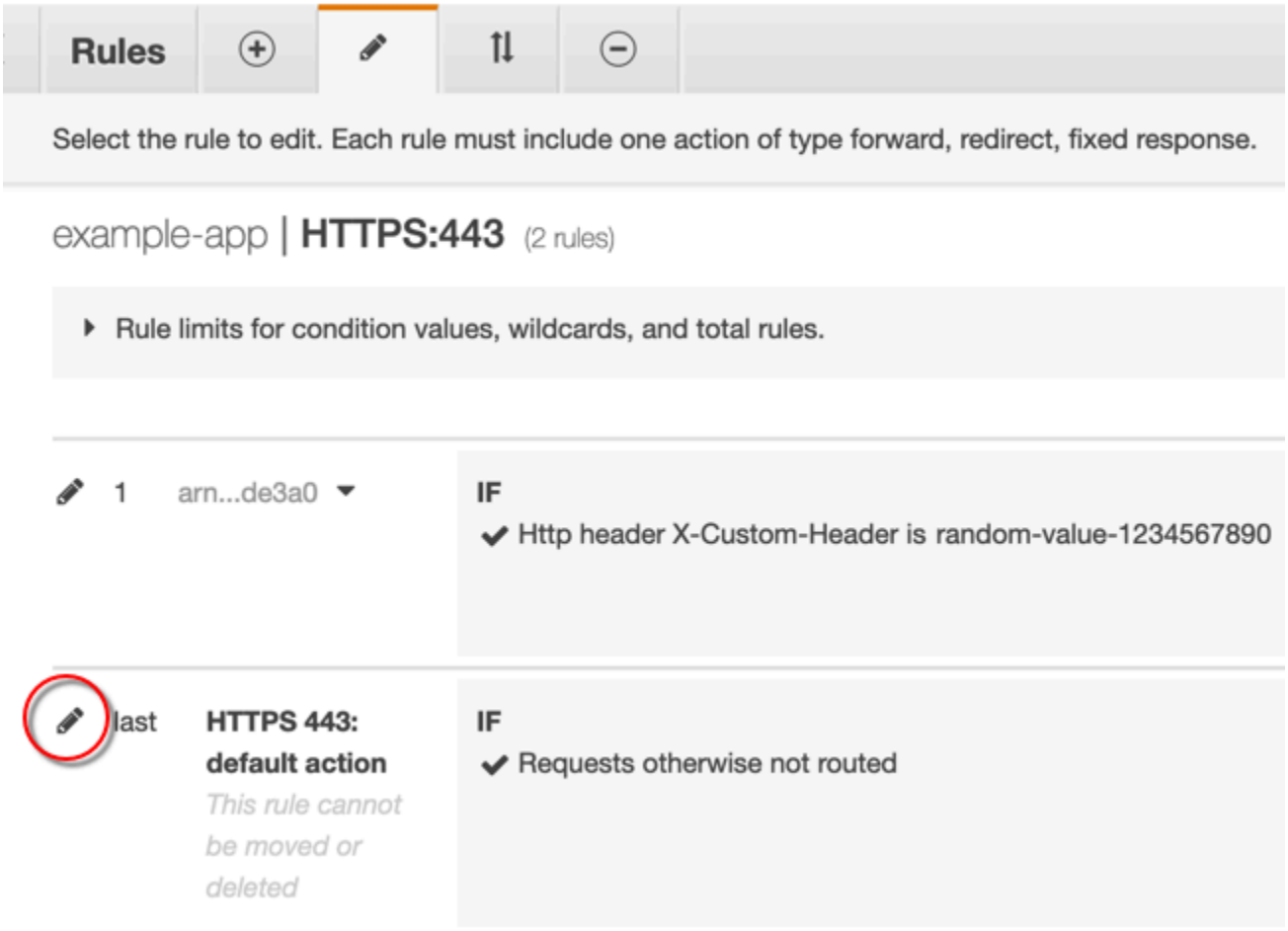
c. [保存] を選択して新しいルールを作成します。



7. アイコンを選択してルールを編集します。



8. デフォルトルールの編集アイコンを選択します。



9. デフォルトルールでは、次の操作を行います。

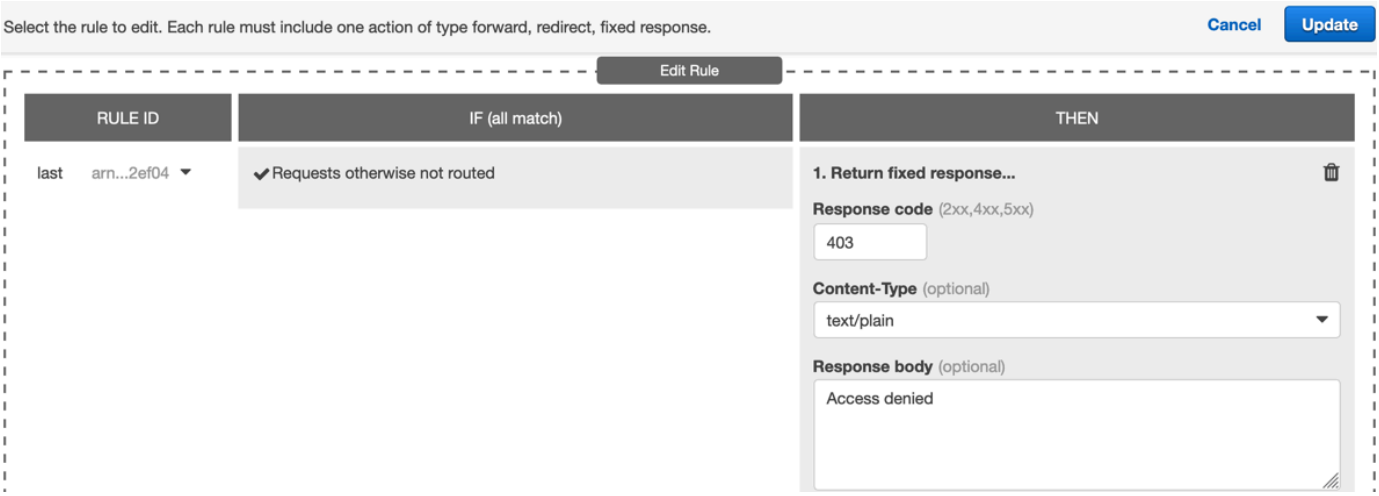
   a. デフォルトのアクションを削除します。

   b. [アクションの追加] を選択してから、[固定応答を返す] を選択します。

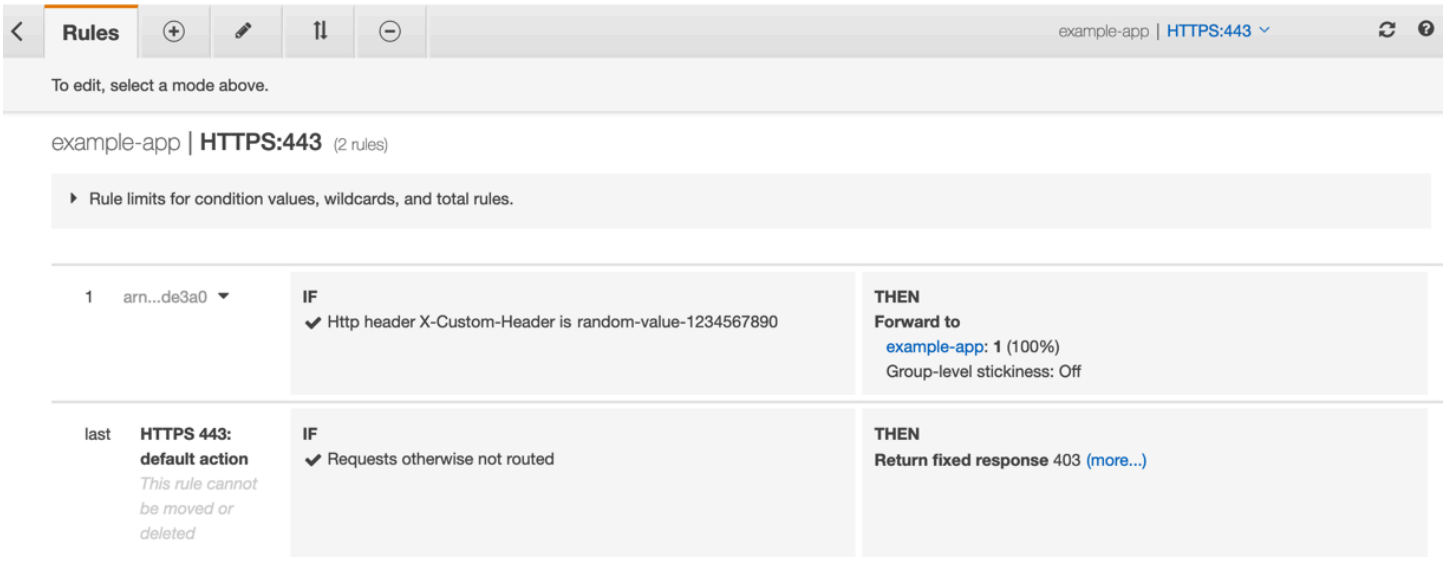
   c. [応答コード] に、**403**を入力します。

   d. [応答本文] に、**Access denied**を入力します 。

   e. 「更新」を選択して、デフォルトルールを更新します。



これらのステップを完了すると、次の図が示すように、ロードバランサーリスナーに 2 つのルールがあります。最初のルールは、HTTP ヘッダーを含むリクエスト (CloudFront からのリクエスト) を転送します。2 番目のルールは、他のすべてのリクエスト (CloudFront 以外からのリクエスト) に対して固定レスポンスを送信します。



CloudFront ディストリビューションと Application Load Balancer にリクエストを送信することで、ソリューションが機能することを確認できます。CloudFront へのリクエストは、ウェブアプリケーションまたはコンテンツを返し、Application Load Balancer に直接送信されたリクエストは、403プレーンテキストメッセージが入ったレスポンスを返しますAccess denied。

（オプション）このソリューションのセキュリティ向上

このソリューションのセキュリティ向上のために、Application Load Balancer にリクエストを送信するときに、常に HTTPS を使用するように CloudFront ディストリビューションを設定できます。このソリューションは、カスタムヘッダー名と値を機密に保つ場合に限り機能します。HTTPS を使用すると、盗聴者がヘッダー名と値を検出するのを防ぐことに役立ちます。また、ヘッダー名と値を定期的に交換することをお勧めします。

オリジンリクエストに HTTPS を使用する

オリジンリクエストに HTTPS を使用するように CloudFront を設定するためには、[オリジンプロトコルポリシー] 設定を [HTTPS だけ] に設定します。この設定は、CloudFront コンソール、AWS CloudFormation、および CloudFront API で使用できます。詳細については、「プロトコル (カスタムオリジンのみ)」を参照してください。

オリジンリクエストで HTTPS を使用するように CloudFront を設定する場合は、以下も適用されます。

- オリジンリクエストポリシーを使用して `Host` ヘッダーをオリジンに転送するように CloudFront を設定する必要があります。AllViewer マネージドオリジンリクエストポリシーを使用できます。
- 前のセクションで示したように、Application Load Balancer に HTTPS リスナーがあることを確認します。詳細については、Application Load Balancer のユーザーガイドで「HTTPS リスナーの作成」を参照してください。HTTPS リスナーを使用するには、Application Load Balancer にルーティングするドメイン名と一致する SSL/TLS 証明書が必要です。
- CloudFront の SSL/TLS 証明書は、AWS Certificate Manager (ACM) の `us-east-1` AWS リージョンでのみリクエスト (またはインポート) できます。CloudFront はグローバルサービスであるため、証明書は、`us-east-1` リージョンから CloudFront ディストリビューションに関連するすべてのリージョンへと自動的に配布されます。
  - 例えば、`ap-southeast-2` リージョンに Application Load Balancer (ALB) がある場合、`ap-southeast-2` リージョン (CloudFront と ALB オリジンの間で HTTPS を使用する場合) と `us-east-1` リージョン (ビューワーと CloudFront の間で HTTPS を使用する場合) の両方で SSL/TLS 証明書を設定する必要があります。どちらの証明書も、Application Load Balancer にルーティングするドメイン名と一致する必要があります。詳細については、「AWS Certificate Manager の場合は AWS リージョン」を参照してください。
- ウェブアプリケーションのエンドユーザー（ 閲覧者、または クライアント とも呼ばれる ）が HTTPS を使用できる場合には、エンドユーザーからの HTTPS 接続を優先（または必要とする）ように CloudFront を構成することもできます。これを行うためには、[ビューワープロトコル

ポリシー] 設定を使用します。エンドユーザを HTTP から HTTPS にリダイレクトする、または HTTP を使用するリクエストを拒否するように設定できます。この設定は、CloudFront コンソール、AWS CloudFormation、および CloudFront API で使用できます。詳細については、「ビューワープロトコルポリシー」を参照してください。

ヘッダー名と値を交換する

HTTPS の使用に加え、ヘッダー名と値を定期的に交換することをお勧めします。これを行うためのハイレベルな手順は次のとおりです。

1. CloudFront を設定して、Application Load Balancer に送信するリクエストに追加のカスタム HTTP ヘッダーを追加します。
2. Application Load Balancer リスナールールを更新して、この追加のカスタム HTTP ヘッダーを含むリクエストを転送します。
3. CloudFront を設定して、Application Load Balancer バランサーに送信するリクエストへの元のカスタム HTTP ヘッダー追加を停止します。
4. Application Load Balancer リスナールールを更新して、元のカスタム HTTP ヘッダーを含むリクエストの転送を停止します。

これらの手順を実行する方法の詳細については、前述のセクションを参照してください。

(オプション) CloudFront の AWS マネージドプレフィックスリストを使用してオリジンへのアクセスを制限します。

Application Load Balancer へのアクセスをさらに制限するには、サービスが AWS マネージドプレフィックスリストを使用しているときに CloudFront からのトラフィックのみを受け入れるように、Application Load Balancer に関連付けたセキュリティグループを設定します。これにより、CloudFront から発信されていないトラフィックは、ネットワーク層 (レイヤー 3) またはトランスポート層 (レイヤー 4) で Application Load Balancer に到達しないようになります。

詳細については、ブログ記事「Amazon CloudFront の AWS マネージドプレフィックスリストを使用してオリジンへのアクセスを制限する」を参照してください。

コンテンツの地理的配分を制限する

地理的制限 (地理的ブロックとも呼ばれます) を使用すると、Amazon CloudFront ディストリビューションを通じて配信しているコンテンツに対して特定地域のユーザーがアクセスできないようにすることができます。地理的制限を使用するには、次の 2 つの方法があります。

- CloudFront の地理的制限機能を使用する。ディストリビューションに関連するすべてのファイルへのアクセスを制限し、国レベルでアクセスを制限する場合は、この方法を使用します。
- サードパーティーの位置情報サービスを使用する。ディストリビューションに関連するファイルのサブセットへのアクセスを制限する場合や、国レベルより詳細なレベルでアクセスを制限する場合は、この方法を使用します。

トピック

CloudFront の地理的制限を使用する

ユーザーがコンテンツをリクエストすると、通常 CloudFront はユーザーがいる場所に関係なくリクエストされたコンテンツを提供します。特定の国のユーザーによるコンテンツへのアクセスを回避する必要がある場合は、CloudFront の地理的制限を使用して、次のいずれかを行うことができます。

- 承認された国の許可リストに含まれているいずれかの国にユーザーがいる場合のみ、コンテンツへのアクセス許可を付与する。
- ユーザーが拒否リストにある禁止国にいる場合、コンテンツへのアクセスを禁止する。

例えば、コンテンツの配信が許可されていない国がリクエスト元である場合は、CloudFront の地理的制限を使用してリクエストをブロックできます。

> **Note**
>
> CloudFront は、サードパーティーのデータベースを使用して、ユーザーがいる場所を判別します。IP アドレスと国とのマッピングの正確さは、リージョンによって異なります。最近のテストによれば、全体的な正確性は 99.8% です。CloudFront がユーザーの場所を特定できない場合、ユーザーがリクエストしたコンテンツは CloudFront から供給されます。

地理的制限は次のような仕組みになっています。

1. 仮に、コンテンツをリヒテンシュタインでのみ配信する権限を持っているとしましょう。CloudFront ディストリビューションを更新して、リヒテンシュタインのみを含む許可リストを追加します。(または、リヒテンシュタイン以外のすべての国を含む拒否リストを追加することもできます)。
2. モナコに住むユーザーからコンテンツがリクエストされた場合、DNS はそのリクエストをミラノ (イタリア) にある CloudFront エッジロケーションにルーティングします。
3. ミラノのエッジロケーションはお客さまのディストリビューションを検索し、モナコ王国のユーザーはコンテンツをダウンロードするアクセス許可がないと判断します。
4. CloudFront は HTTP ステータスコード `403 (Forbidden)` をユーザーに返します。

オプションで、ユーザーにカスタムエラーメッセージを返すよう CloudFront を設定することも、リクエストされたファイルに関するエラーレスポンスを CloudFront でキャッシュしておく時間の長さを指定することもできます。デフォルト値は 10 秒です。詳細については、「HTTP ステータスコード別のカスタムエラーページを作成する」を参照してください。

地理的制限はディストリビューション全体に適用されます。コンテンツのある部分に特定の制限を適用し、コンテンツの別の部分に別の制限を適用する (または制限を適用しない) 必要がある場合は、別々の CloudFront ディストリビューションを作成するか、サードパーティの位置情報サービスを使用する必要があります。

CloudFront の標準ログ (アクセスログ) を有効にすると、`sc-status` (HTTP ステータスコード) の値が `403` であるログエントリを検索することによって、CloudFront で拒否されたリクエストを特定できます。ただし、標準ログだけでは、ユーザーのいる場所に基づいて CloudFront が拒否したリクエストと、別の理由でファイルへのアクセス許可がユーザーになかったために CloudFront が拒否したリクエストとを区別することができません。Digital Element や MaxMind などサードパーティーの位置情報サービスを利用している場合は、アクセスログの `c-ip` (クライアント IP) 列にある IP アドレスに基づいてリクエストの場所を識別できます。CloudFront の標準ログの詳細については、標準ログ (アクセスログ) の設定および使用 を参照してください。

次の手順では、CloudFront コンソールを使用して、地理的制限を既存のディストリビューションに追加する方法を示します。コンソールを使用してディストリビューションを作成する方法の詳細については、「ディストリビューションを作成する」を参照してください。

CloudFront ウェブディストリビューションに地理的制限を追加するには (コンソール)

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[ディストリビューション] を選択し、更新するディストリビューションを選択します。
3. [セキュリティ] タブを選択し、[地理的制限] を選択します。
4. [編集] を選択します。
5. 許可した国のリストを作成する場合は、許可リスト、またはブロックした国のリストを作成する場合は、ブロックリストを作成します。
6. 目的の国をリストに追加し、[Save changes] (変更の保存) を選択します。

サードパーティの位置情報サービスを使用する

CloudFront の地理的制限機能を使用すると、特定のウェブディストリビューションで配信するすべてのファイルについて、国レベルでコンテンツの配信を制御できます。地理的制限が国境と一致していない場合や、特定のディストリビューションで配信するファイルの一部にのみアクセスを制限する必要がある場合は、CloudFront とサードパーティーの位置情報サービスを組み合わせることができます。これにより、国だけではなく、都市、郵便番号、または緯度/経度に基づいてコンテンツの制御を提供します。

サードパーティーの位置情報サービスを使用する場合は、有効期限の日時を指定できる CloudFront 署名付き URL を使用することをお勧めします。有効期限を経過すると、この URL は無効になります。さらに、オリジンとして Amazon S3 バケットを使用することをお勧めします。CloudFront オリジンアクセスコントロールを使用することで、オリジンのコンテンツへのユーザーによる直接アクセスを回避できるためです。署名付き URL とオリジンアクセスコントロールの詳細については、「署名付き URL と署名付き Cookie を使用したプライベートコンテンツを提供する」を参照してください。

以下のステップは、サードパーティーの位置情報サービスを使用してファイルへのアクセスを制御する方法を説明しています。

サードパーティーの位置情報サービスを使用して CloudFront ディストリビューション内のファイルへのアクセスを制限するには

1. 位置情報サービスのアカウントを取得します。
2. コンテンツを Amazon S3 バケットにアップロードします。

3. プライベートコンテンツを供給できるように Amazon CloudFront と Amazon S3 を設定します。詳細については、「署名付き URL と署名付き Cookie を使用したプライベートコンテンツを提供する」を参照してください。
4. 以下の処理を行うようにウェブアプリケーションを記述します。

   - 各ユーザーリクエストの IP アドレスを位置情報サービスに送信します。
   - 位置情報サービスからの戻り値を評価し、ユーザーの場所が CloudFront によるコンテンツの配信先に該当するかどうかを判別します。
   - コンテンツをユーザーの場所に配信する場合、CloudFront コンテンツの署名付き URL を生成します。コンテンツをその場所に配信しない場合、HTTP ステータスコード 403 (Forbidden) をユーザーに返します。または、カスタムエラーメッセージが返されるように CloudFront を設定することもできます。詳細については、「the section called “HTTP ステータスコード別のカスタムエラーページを作成する”」を参照してください。

   詳細については、使用する位置情報サービスのドキュメントを参照してください。

ウェブサーバー変数を使用すると、ウェブサイトを訪れたユーザーの IP アドレスを取得できます。次の点に注意してください。

- ウェブサーバーがインターネットにロードバランサー経由で接続されていない場合、ウェブサーバー変数を使用してリモート IP アドレスを取得できます。ただし、この IP アドレスが必ずしもユーザーの IP アドレスであるとは限りません。ユーザーのインターネットへの接続方法によっては、プロキシサーバーの IP アドレスである可能性もあります。
- ウェブサーバーがインターネットにロードバランサー経由で接続されている場合、ウェブサーバー変数には、ユーザーの IP アドレスではなく、ロードバランサーの IP アドレスが含まれる可能性があります。この構成では、X-Forwarded-For HTTP ヘッダーに含まれる最後の IP アドレスを使用することをお勧めします。通常、このヘッダーには複数の IP アドレスが含まれており、そのほとんどはプロキシまたはロードバランサーの IP アドレスです。ユーザーの地理的な場所に関連付けられている可能性が最も高い IP アドレスは、リストの最後にある IP アドレスです。

ウェブサーバーがロードバランサーに接続されていない場合は、IP アドレスのスプーフィングを回避するために、X-Forwarded-For ヘッダーではなくウェブサーバー変数を使用することをお勧めします。

フィールドレベル暗号化を使用した機密データの保護

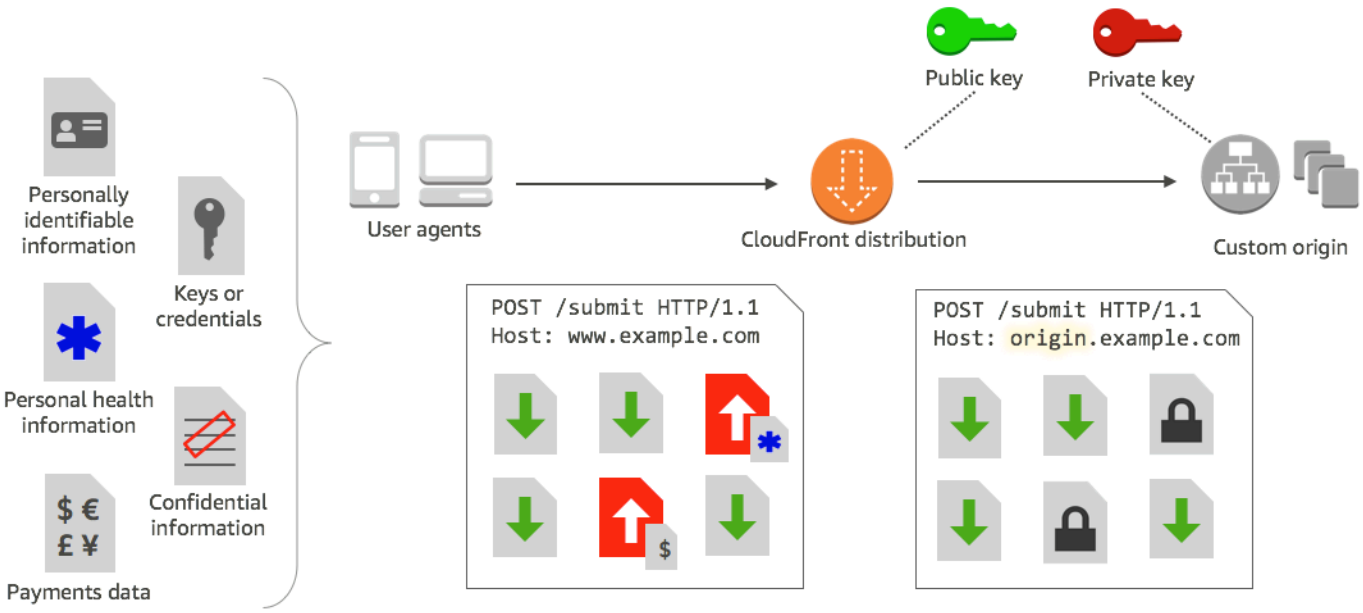
Amazon CloudFront では、HTTPS を使用して、オリジンサーバーへの安全なエンドツーエンド接続を強制できます。フィールドレベル暗号化では、セキュリティのレイヤーが追加されます。これにより、システムの処理中に特定のデータに特定のアプリケーションのみがアクセスできるように、そのデータを保護できます。

フィールドレベル暗号化により、ユーザーが機密情報をウェブサーバーに安全にアップロードできるようになります。ユーザーから提供される機密情報は、ユーザー近くのエッジで暗号化され、アプリケーションスタック全体で暗号化された状態が維持されます。この暗号化により、データを必要としており、復号するための認証情報を持つアプリケーションだけが暗号化できるようになります。

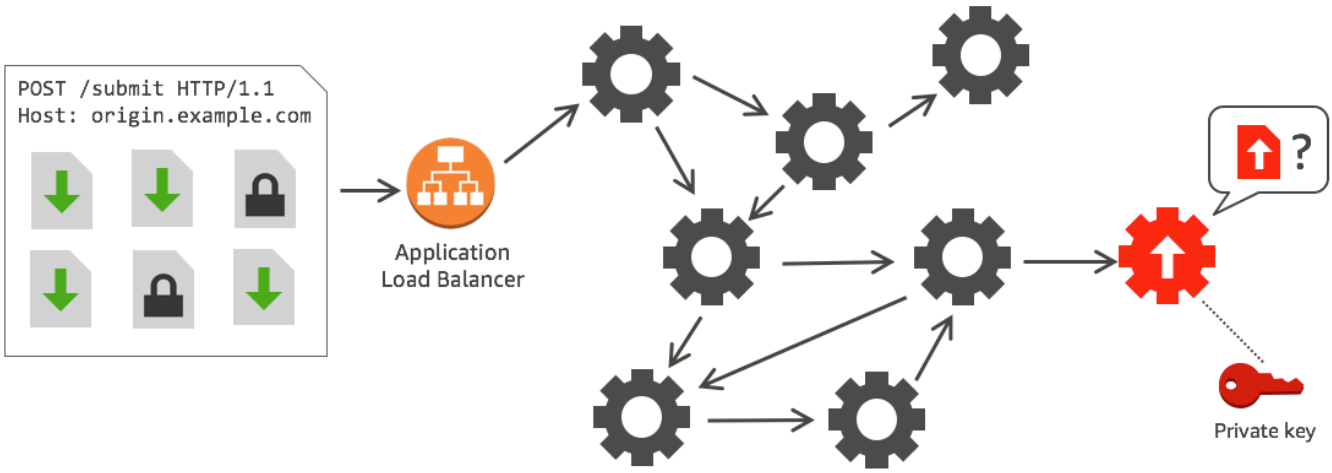
フィールドレベル暗号化を使用するには、CloudFront ディストリビューションの設定で、暗号化する POST リクエストの一連のフィールドと、それらの暗号化に使用するパブリックキーを指定します。リクエストの最大 10 個のデータフィールドを暗号化できます (フィールドレベル暗号化を使用して、リクエストのすべてのデータをまとめて暗号化することはできません。暗号化する個々のフィールドを指定する必要があります)。

フィールドレベル暗号化を含む HTTPS リクエストがオリジンに転送され、リクエストがオリジンアプリケーションまたはサブシステム全体にルーティングされても、機密データは暗号化されたままです。それにより、機密データの漏洩や偶発的な損失のリスクが軽減されます。クレジット番号へのアクセスを必要とする支払い処理システムなど、ビジネス上の理由で機密データにアクセスする必要のあるコンポーネントは、適切なプライベートキーを使用してデータを復号化し、そのデータにアクセスできます。

> **Note**
>
> フィールドレベル暗号化を使用するには、オリジンがチャンクエンコードをサポートしている必要があります。



CloudFront のフィールドレベル暗号化では、非対称暗号化 (パブリックキー暗号化とも呼ばれる) が使用されます。CloudFront にパブリックキーを渡すと、指定したすべての機密データが自動的に暗号化されます。CloudFront に渡したキーは、暗号化された値の復号化には使用できません。この目的で使用できるのはプライベートキーのみです。



トピック

- フィールドレベル暗号化の概要
- フィールドレベル暗号化を設定する
- オリジンでデータフィールドを復号する

フィールドレベル暗号化の概要

以下に示しているのは、フィールドレベル暗号化の設定手順の概要です。この手順の詳細については、「フィールドレベル暗号化を設定する」を参照してください。

1. パブリックキーとプライベートキーのペアを取得する。CloudFront でフィールドレベル暗号化を設定する前に、パブリックキーを取得して追加する必要があります。
2. フィールドレベル暗号化のプロファイルを作成する。CloudFront で作成するフィールドレベル暗号化のプロファイルで、暗号化するフィールドを定義します。
3. フィールドレベル暗号化の設定を作成する。設定では、特定のデータフィールドの暗号化に使用するプロファイル (リクエストのコンテンツタイプまたはクエリ引数に基づく) を指定します。また、さまざまなシナリオで必要なリクエスト転送動作オプションを選択することもできます。例えば、リクエスト URL のクエリ引数で指定されたプロファイル名が CloudFront に存在しない場合の動作を設定できます。
4. キャッシュ動作にリンクする。設定をディストリビューションのキャッシュ動作にリンクして、いつ CloudFront がデータを暗号化するかを指定します。

フィールドレベル暗号化を設定する

フィールドレベル暗号化を使用するには、以下の手順に従います。フィールドレベル暗号化のクォータ (以前は制限と呼ばれていました) の詳細については、「クォータ」を参照してください。

- ステップ 1: RSA キーペアを作成する
- ステップ 2: パブリックキーを CloudFront に追加する
- ステップ 3: フィールドレベル暗号化のプロファイルを作成する
- ステップ 4: 設定を作成する
- ステップ 5: キャッシュ動作に設定を追加する

ステップ 1: RSA キーペアを作成する

開始するには、パブリックキーとプライベートキーを含む RSA キーペアを作成する必要があります。パブリックキーにより CloudFront がデータを暗号化できるようになり、プライベートキーによりオリジンのコンポーネントが暗号化されたフィールドを復号できるようになります。OpenSSL または別のツールを使用してキーペアを作成できます。キーのサイズは 2,048 ビットであることが必要です。

たとえば、OpenSSL を使用する場合、次のコマンドを使用して 2048 ビット長のキーペアを生成し、private_key.pem ファイルに保存できます。

```
openssl genrsa -out private_key.pem 2048
```

生成されるファイルには、パブリックキーとプライベートキーの両方が含まれます。そのファイルからパブリックキーを抽出するには、次のコマンドを実行します。

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

公開キーファイル (public_key.pem) には、次のステップで貼り付けるエンコードされたキー値が含まれています。

ステップ 2: パブリックキーを CloudFront に追加する

RSA キーペアを取得したら、パブリックキーを CloudFront に追加します。

パブリックキーを CloudFront に追加するには (コンソール)

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[Public key (パブリックキー)] を選択します。
3. [Add public key (パブリックキーを追加)] を選択します。
4. [Key name (キー名)] にキーの一意の名前を入力します。キー名にはスペースを使用できません。英数字、アンダースコア (_)、ハイフン (-) のみを使用できます。最大長は 128 文字です。
5. [Key value (キー値)] に、-----BEGIN PUBLIC KEY----- および -----END PUBLIC KEY----- 行を含む、パブリックキーのエンコードされたキー値を貼り付けます。
6. [Comment (コメント)] で、オプションのコメントを追加します。たとえば、パブリックキーの有効期限を含めることができます。
7. 追加 を選択します。

CloudFront で使用するキーを追加するには、このステップの手順を繰り返します。

ステップ 3: フィールドレベル暗号化のプロファイルを作成する

1 つ以上のパブリックキーを CloudFront に追加した後、CloudFront にどのフィールドを暗号化するかを指示するプロファイルを作成します。

フィールドレベル暗号化のプロファイルを作成するには (コンソール)

1. ナビゲーションペインで、[Field-level encryption (フィールドレベル暗号化)] を選択します。
2. [Create profile (プロファイルの作成)] を選択します。
3. 以下のフィールドに入力します。

   Profile name

   プロファイルの一意の名前を入力します。キー名にはスペースを使用できません。英数字、アンダースコア (_)、ハイフン (-) のみを使用できます。最大長は 128 文字です。

   Public key name

   ドロップダウンリストから、ステップ 2 で CloudFront に追加したパブリックキーの名前を選択します。CloudFront では、このプロファイルで指定したフィールドの暗号化に、このキーが使用されます。

   Provider name

   キーを識別するのに役立つフレーズ (キーペアを取得したプロバイダーなど) を入力します。この情報は、プライベートキーと共に、アプリケーションがデータフィールドを復号化するときに必要になります。プロバイダー名にはスペースを使用できません。英数字、コロン (:)、アンダースコア (_)、ハイフン (-) のみを使用できます。最大長は 128 文字です。

   Field name pattern to match

   CloudFront で暗号化するデータフィールドの名前、またはリクエスト内のデータフィールド名を識別するパターンを入力します。+ オプションを選択して、このキーで暗号化するすべてのフィールドを追加します。

   フィールド名パターンには、データフィールドの名前全体を入力するか (DateOfBirth など)、ワイルドカード文字 (*) を使用して名前の最初の部分のみを入力します (CreditCard* など)。フィールド名パターンには、オプションのワイルドカード文字 (*) に加えて、英数字、角かっこ ([および])、ピリオド (.)、アンダースコア (_)、ハイフン (-) のみを含める必要があります。

   異なるフィールド名パターンに重複する文字を使用しないでください。たとえば、ABC * というフィールド名パターンがある場合、AB * という別のフィールド名パターンを追加することはできません。また、フィールド名は大文字と小文字が区別され、最大長は 128 文字です。

コメント

(オプション) このプロファイルに関するコメントを入力します。最大長は 128 文字です。

4. フィールドに入力した後、[Create profile (プロファイルの作成)] を選択します。
5. さらにプロファイルを追加する場合は、[Add profile (プロファイルの追加)] を選択します。

ステップ 4: 設定を作成する

1 つ以上のフィールドレベル暗号化のプロファイルを作成した後、リクエストのコンテンツタイプを指定する設定を作成します。この設定には、暗号化するデータ、暗号化に使用するプロファイル、および CloudFront による暗号化の処理方法を指定するその他のオプションが含まれます。

たとえば、CloudFront がデータを暗号化できないときに、CloudFront が以下のシナリオでリクエストをブロックするかオリジンに転送するかを指定できます。

- リクエストのコンテンツタイプが設定にない場合 – コンテンツタイプが設定に追加されていない場合に、CloudFront がそのコンテンツタイプのリクエストをデータフィールドの暗号化なしでオリジンに転送するか、ブロックしてエラーを返すかを指定できます。

  > **Note**
  >
  > コンテンツタイプを設定に追加していても、そのタイプで使用するプロファイルを指定していない場合、そのコンテンツタイプのリクエストは、CloudFront により常にオリジンに転送されます。

- クエリ引数で指定したプロファイル名が不明な場合 – `fle-profile` クエリ引数で、ディストリビューションに存在しないプロファイル名が指定されている場合に、CloudFront がリクエストをデータフィールドの暗号化なしでオリジンに送るか、リクエストをブロックしてエラーを返すかを指定できます。

設定で、URL でクエリ引数としてプロファイルを提供するかどうかを指定して、そのクエリのコンテンツタイプにマッピングしたプロファイルを上書きすることもできます。デフォルトでは、CloudFront では、コンテンツタイプにマッピングしたプロファイル (指定してある場合) が使用されます。これにより、デフォルトで使用されるプロファイルを提供できますが、特定のリクエストに使用される別のプロファイルを指定することもできます。

たとえば、使用するクエリ引数プロファイルとして (設定で) **`SampleProfile`** を指定できます。さらに、`https://d1234.cloudfront.net?fle-profile=SampleProfile` の代わりに URL

`https://d1234.cloudfront.net` を使用できます。これにより、CloudFront がこのリクエストに、リクエストのコンテンツタイプ用に設定したプロファイルではなく **`SampleProfile`** を使用できるようになります。

1 つのアカウント用に最大 10 個の設定を作成し、いずれかの設定をそのアカウントの任意のディストリビューションのキャッシュ動作に関連付けることができます。

フィールドレベル暗号化の設定を作成するには (コンソール)

1. [Field-level encryption (フィールドレベル暗号化)] ページで、[Create configuration (設定の作成)] を選択します。

   注意: プロファイルを 1 つ以上作成していない場合、設定を作成するオプションは表示されません。

2. 以下のフィールドに入力して、使用するプロファイルを指定します (一部のフィールドは変更できません)。

   Content type (変更不可)

   コンテンツタイプは `application/x-www-form-urlencoded` に設定されており、変更することはできません。

   Default profile ID (オプション)

   ドロップダウンリストから、[Content type (コンテンツタイプ)] フィールドでコンテンツタイプにマッピングするプロファイルを選択します。

   Content format (変更不可)

   コンテンツ形式は `URLencoded` に設定されており、変更することはできません。

3. 以下のオプションに対する CloudFront のデフォルト動作を変更する場合は、該当するチェックボックスをオンにします。

   Forward request to origin when request's content type is not configured

   リクエストのコンテンツタイプに使用するプロファイルを指定していないときに、リクエストをオリジンに転送させる場合は、このチェックボックスをオンにします。

   Override the profile for a content type with a provided query argument

   クエリ引数で指定したプロファイルによって、コンテンツタイプ用に指定したプロファイルを上書する場合は、このチェックボックスをオンにします。

4. このチェックボックスをオンにして、クエリ引数によってデフォルトプロファイルが上書きされるようにした場合は、設定で以下の追加フィールドを入力する必要があります。クエリで使用するこれらのクエリ引数マッピングは最大 5 つ作成できます。

   Query argument

   `fle-profile` クエリ引数の URL に含める値を入力します。CloudFront は、このクエリ引数の値に関連付けられたプロファイル ID (次のフィールドで指定) をこのクエリのフィールドレベル暗号化に使用します。

   最大長は 128 文字です。値にはスペースを含めることはできません。英数字、ダッシュ (-)、ピリオド (.)、アンダースコア (_)、アスタリスク (*)、プラス記号 (+)、パーセント (%) のみを使用する必要があります。

   Profile ID

   ドロップダウンリストから、[Query argument (クエリ引数)] に入力した値に関連付けるプロファイルを選択します。

   Forward request to origin when the profile specified in a query argument does not exist

   クエリ引数で指定したプロファイルが CloudFront で定義されていないときに、リクエストがオリジンに転送されるようにするには、このチェックボックスをオンにします。

ステップ 5: キャッシュ動作に設定を追加する

フィールドレベル暗号化を使用するには、設定 ID をディストリビューション用の値として追加して、ディストリビューションのキャッシュ動作に設定をリンクします。

> **Important**
>
> フィールドレベルの暗号化設定をキャッシュ動作にリンクするには、常に HTTPS を使用し、ビューワーからの HTTP POST および PUT リクエストを受け入れるようにディストリビューションを設定する必要があります。つまり、以下が満たされている必要があります。
>
> - キャッシュ動作の [ビューワープロトコルポリシーは] が [HTTP を HTTPS にリダイレクト] または [HTTPS のみ] に設定されている必要があります。(AWS CloudFormation または CloudFront API で、`ViewerProtocolPolicy` は `redirect-to-https` または `https-only` に設定されている必要があります)。

- キャッシュ動作の [Allowed HTTP Methods] (許可された HTTP メソッド) が、[GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE] に設定されている必要があります。(AWS CloudFormation または CloudFront API では、`AllowedMethods` が `GET、HEAD、OPTIONS、PUT、POST、PATCH、DELETE` に設定されている必要があります。これらは任意の順序で指定できます)。
- オリジン設定の [オリジンプロトコルポリシー] が [ビューワーに合わせる] または [HTTPS のみ] に設定されている必要があります。(AWS CloudFormation または CloudFront API で、`OriginProtocolPolicy` は `match-viewer` または `https-only` に設定されている必要があります)。

詳細については、「ディストリビューション設定リファレンス」を参照してください。

オリジンでデータフィールドを復号する

CloudFront は AWS Encryption SDK を使用してデータフィールドを暗号化します。データはアプリケーションスタック全体で暗号化されたままであり、復号化するための認証情報のあるアプリケーションによってのみアクセスできます。

暗号化の後、暗号テキストは base64 でエンコードされます。アプリケーションがオリジンでテキストを復号化するときには、まず暗号テキストをデコードしてから、AWS Encryption SDK を使用してデータを復号化する必要があります。

以下のサンプルコードでは、アプリケーションがオリジンでデータを復号化する方法を示しています。次の点に注意してください。

- わかりやすい例になるように、このサンプルは作業ディレクトリのファイルからパブリックキーとプライベートキー (DER 形式) を読み込みます。実際には、オフラインのハードウェアセキュリティモジュールなどの安全なオフラインの場所にプライベートキーを保存し、開発チームにパブリックキーを配布します。
- CloudFront はデータの暗号化時に特定の情報を使用します。データの復号化には、オリジンで同じ一連のパラメータを使用する必要があります。以下に示しているのは、MasterKey の初期化時に CloudFront が使用するパラメータです。
  - PROVIDER_NAME: フィールドレベル暗号化のプロファイルを作成したときにこの値を指定しました。同じ値をここで使用します。
  - KEY_NAME: パブリックキーを CloudFront にアップロードしたときに作成し、プロファイルで指定したキー名です。同じ値をここで使用します。

- ALGORITHM: CloudFront では暗号化アルゴリズムとして RSA/ECB/OAEPWithSHA-256AndMGF1Padding が使用されるため、データの復号化には同じアルゴリズムを使用する必要があります。

- 暗号テキストを入力として以下のサンプルプログラムを実行すると、復号化されたデータがコンソールに出力されます。詳細については、Encryption SDK の「Java サンプルコードAWS」を参照してください。

サンプルコード

```
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;

import org.apache.commons.codec.binary.Base64;

import com.amazonaws.encryptionsdk.AwsCrypto;
import com.amazonaws.encryptionsdk.CryptoResult;
import com.amazonaws.encryptionsdk.jce.JceMasterKey;

/**
 * Sample example of decrypting data that has been encrypted by CloudFront field-level
 encryption.
 */
public class DecryptExample {

    private static final String PRIVATE_KEY_FILENAME = "private_key.der";
    private static final String PUBLIC_KEY_FILENAME = "public_key.der";
    private static PublicKey publicKey;
    private static PrivateKey privateKey;

    // CloudFront uses the following values to encrypt data, and your origin must use
 same values to decrypt it.
    // In your own code, for PROVIDER_NAME, use the provider name that you specified
 when you created your field-level
    // encryption profile. This sample uses 'DEMO' for the value.
    private static final String PROVIDER_NAME = "DEMO";
```

```
    // In your own code, use the key name that you specified when you added your public
key to CloudFront. This sample
    // uses 'DEMOKEY' for the key name.
    private static final String KEY_NAME = "DEMOKEY";
    // CloudFront uses this algorithm when encrypting data.
    private static final String ALGORITHM = "RSA/ECB/OAEPWithSHA-256AndMGF1Padding";

    public static void main(final String[] args) throws Exception {

        final String dataToDecrypt = args[0];

        // This sample uses files to get public and private keys.
        // In practice, you should distribute the public key and save the private key
in secure storage.
        populateKeyPair();

        System.out.println(decrypt(debase64(dataToDecrypt)));
    }

    private static String decrypt(final byte[] bytesToDecrypt) throws Exception {
        // You can decrypt the stream only by using the private key.

        // 1. Instantiate the SDK
        final AwsCrypto crypto = new AwsCrypto();

        // 2. Instantiate a JCE master key
        final JceMasterKey masterKey = JceMasterKey.getInstance(
                publicKey,
                privateKey,
                PROVIDER_NAME,
                KEY_NAME,
                ALGORITHM);

        // 3. Decrypt the data
        final CryptoResult <byte[], ? > result = crypto.decryptData(masterKey,
bytesToDecrypt);
        return new String(result.getResult());
    }

    // Function to decode base64 cipher text.
    private static byte[] debase64(final String value) {
        return Base64.decodeBase64(value.getBytes());
    }
```

```
    private static void populateKeyPair() throws Exception {
        final byte[] PublicKeyBytes =
 Files.readAllBytes(Paths.get(PUBLIC_KEY_FILENAME));
        final byte[] privateKeyBytes =
 Files.readAllBytes(Paths.get(PRIVATE_KEY_FILENAME));
        publicKey = KeyFactory.getInstance("RSA").generatePublic(new
 X509EncodedKeySpec(PublicKeyBytes));
        privateKey = KeyFactory.getInstance("RSA").generatePrivate(new
 PKCS8EncodedKeySpec(privateKeyBytes));
    }
}
```

CloudFront を使用したビデオオンデマンドおよびライブストリーミングビデオ

CloudFront では、任意の HTTP オリジンを使用して、ビデオオンデマンド (VOD) またはライブストリーミングビデオの配信が可能です。クラウドでビデオワークフローを設定する方法の 1 つは、CloudFront をAWS Media Services と共に使用することです。

トピック

ストリーミングビデオについて

CloudFront がビデオコンテンツを配信できるようにするには、エンコーダーを使用してそのコンテンツをパッケージ化する必要があります。パッケージ化プロセスでは、オーディオ、ビデオ、キャプションのコンテンツを含むセグメントが作成されます。また、マニフェストファイルを生成します。マニフェストファイルは、どのセグメントをいつ再生するかについての特定の順序を説明します。パッケージの一般的な形式は MPEG DASH、Apple HLS、Microsoft Smooth Streaming、CMAF です。

VOD ストリーミング

VOD ストリーミングの場合、ビデオコンテンツはサーバーに保存され、ビューワーはいつでも視聴できます。ビューワーがストリーミングできるアセットを作成するには AWS Elemental MediaConvert などのエンコーダを使用してメディアファイルをフォーマットおよびパッケージ化します。

ビデオを適切な形式に変換した後、サーバーまたは Amazon S3 バケットに保存したら、ビューワーのリクエストに応じて CloudFront で配信できます。

ライブビデオストリーミング

ライブビデオストリーミングの場合、ビデオコンテンツは、ライブイベントが発生するとリアルタイムでストリーミングされるか、24 時間 365 日のライブチャンネルとして設定されます。ブロードキャストおよびストリーミング配信用のライブ出力を作成するには、AWS Elemental

MediaLive などのエンコーダを使用してビデオを圧縮し、デバイス表示用にフォーマットします。

ビデオがエンコードされたら、そのビデオを AWS Elemental MediaStore に保存するか、AWS Elemental MediaPackage を使用してさまざまな配信形式に変換できます。これらのオリジンを使用し、CloudFront ディストリビューションを設定して、コンテンツを配信できます。これらのサービスと連携するディストリビューションを作成するための具体的な手順とガイダンスについては、「AWS Elemental MediaStore をオリジンとして使用してビデオを配信する」と「AWS Elemental MediaPackage でフォーマットされたライブ動画を配信する」を参照してください。

Wowza と統合ストリーミングには、CloudFront でのビデオのストリーミングに使用できるツールも用意されています。CloudFront で Wowza を使用する方法の詳細については、Wowza ドキュメントウェブサイトにある「Wowza Streaming Engine ライセンスを CloudFront ライブ HTTP ストリーミングに持ち込む」を参照してください。VOD ストリーミングのために Unified Streaming を CloudFront で使用する方法については、Unified Streaming のドキュメントウェブサイトの CloudFront を参照してください。

CloudFront でビデオオンデマンドを配信する

CloudFront でビデオオンデマンド (VOD) ストリーミングを配信するには、以下のサービスを使用します。

- Amazon S3 を使用してコンテンツを元の形式で保存し、トランスコードされたビデオを保存します。
- ビデオをストリーミング形式に変換するエンコーダ (AWS Elemental MediaConvert など) 。
- CloudFront を使用して、トランスコードされたビデオをビューワーに配信します。Microsoft スムーズストリーミングについては、「Microsoft Smooth Streaming のビデオオンデマンドを設定する」を参照してください。

CloudFront で VOD ソリューションを作成するには

1. コンテンツを Amazon S3 バケットにアップロードします。Amazon S3 の使用の詳細については、Amazon Simple Storage Service ユーザーガイドを参照してください。
2. MediaConvert ジョブを使用してコンテンツをトランスコードします。このジョブは、ビデオをビューワーが使用するプレーヤーが必要とする形式に変換します。ジョブを使用して、解像度とビットレートが異なるアセットを作成することもできます。これらのアセットは、アダプティ

ブビットレート (ABR) ストリーミングに使用され、ビューワーの利用可能な帯域幅に応じて表示品質を調整します。MediaConvert はトランスコードされたビデオを S3 バケットに保存します。

3. CloudFront ディストリビューションを使用して、変換したコンテンツを配信します。ビューワーはいつでも、どのデバイスでもコンテンツを視聴できます。

> **Tip**
>
> AWS CloudFormation テンプレートを使用して VOD AWS ソリューションを関連するすべてのコンポーネントと共にデプロイする方法を検討することができます。テンプレートの使用手順については、AWS オンデマンドビデオガイドの「Automated Deployment (自動デプロイ)」を参照してください。

Microsoft Smooth Streaming のビデオオンデマンドを設定する

CloudFront を使用して、Microsoft Smooth Streaming 形式にトランスコードしたビデオオンデマンド (VOD) コンテンツを配信するには、以下のオプションがあります。

- Microsoft IIS を実行し、ディストリビューションのオリジンとしてスムーズストリーミングをサポートするウェブサーバーを指定します。
- CloudFront ディストリビューションのキャッシュ動作でスムーズストリーミングを有効にします。1 つのディストリビューションでは複数のキャッシュビヘイビアを使用できるため、1 つのディストリビューションをスムーズストリーミングメディアファイルとその他のコンテンツに使用できます。

> **Important**
>
> オリジンとして Microsoft IIS を実行しているウェブサーバーを指定する場合は、CloudFront ディストリビューションのキャッシュ動作でスムーズストリーミングを有効にしないでください。キャッシュ動作としてスムーズストリーミングを有効にした場合、CloudFront は Microsoft IIS サーバーをオリジンとして使用できません。

オリジンサーバーに対してスムーズストリーミングを有効にした場合 (つまり、Microsoft IIS サーバーがない場合) は、以下の点に注意してください。

- 他のコンテンツがキャッシュ動作の [Path Pattern (パスパターン)] の値と一致している場合、同じキャッシュ動作を使用してそのコンテンツも配信できます。
- CloudFront は、スムーズストリーミングメディアファイルで Amazon S3 バケットまたはカスタムオリジンを使用できます。キャッシュ動作に対してスムーズストリーミングを有効にした場合、CloudFront は Microsoft IIS Server をオリジンとして使用できません。
- スムーズストリーミング形式のメディアファイルを無効にすることはできません。有効期限が切れる前にファイルを更新する場合は、ファイルの名前を変更する必要があります。詳細については、「CloudFront が配信するコンテンツを追加、削除、または置き換える」を参照してください。

スムーズストリーミングクライアントについては、Microsoft ドキュメントウェブサイトの「スムーズストリーミング」を参照してください。

Microsoft IIS ウェブサーバーがオリジンでない場合に CloudFront を使用してスムーズストリーミングファイルを配布するには

1. スムーズストリーミングでフラグメント化された MP4 形式にメディアファイルを変換します。
2. 次のいずれかを行ってください。
   - CloudFront コンソールを使用している場合: ディストリビューションを作成または更新するときに、ディストリビューションの 1 つ以上のキャッシュ動作でスムーズストリーミングを有効にします。
   - CloudFront API を使用している場合: ディストリビューションのキャッシュ動作の 1 つ以上の `SmoothStreaming` 複合型に `DistributionConfig` 要素を追加します。
3. スムーズストリーミングファイルをオリジンにアップロードします。
4. `clientaccesspolicy.xml` または `crossdomainpolicy.xml` ファイルを作成し、それをディストリビューションのルートのアクセスできる場所に追加します (例: `https://d111111abcdef8.cloudfront.net/clientaccesspolicy.xml`)。次に、ポリシーの例を示します。

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
<cross-domain-access>
<policy>
<allow-from http-request-headers="*">
<domain uri="*"/>
</allow-from>
<grant-to>
```

```
<resource path="/" include-subpaths="true"/>
</grant-to>
</policy>
</cross-domain-access>
</access-policy>
```

詳細については、Microsoft デベロッパーネットワークウェブサイトの「Making a Service Available Across Domain Boundaries」を参照してください。

5. アプリケーション (メディアプレーヤーなど) 内のリンクの場合は、以下の標準形式でメディアファイルの URL を指定します。

   ```
   https://d111111abcdef8.cloudfront.net/video/presentation.ism/Manifest
   ```

CloudFront と AWS Media Services でライブストリーミングビデオを配信する

CloudFront と AWS Media Services を使用して世界中の視聴者にライブコンテンツを配信するには、次のガイダンスを参照してください。

AWS Elemental MediaLive では、ライブビデオストリームがリアルタイムでエンコードされます。大きいビデオストリームをエンコードするために、MediaLive ではビューワーに配信できる小さいバージョン (エンコード) に圧縮されます。

ライブビデオストリームを圧縮した後、次の 2 つの主要なオプションのいずれかを使用して、コンテンツを準備および配信できます。

- 必要な形式にコンテンツを変換して配信する – ビデオコンテンツが複数の形式で必要な場合は、AWS Elemental MediaPackage を使用して、デバイスタイプ別にコンテンツをパッケージ化できます。コンテンツをパッケージ化するときに、追加の機能を実装し、デジタル著作権管理 (DRM) を追加して、コンテンツの不正使用を防ぐこともできます。CloudFront を使用して MediaPackage でフォーマットされたコンテンツを配信する詳細な手順については、「AWS Elemental MediaPackage でフォーマットされたライブ動画を配信する」を参照してください。
- スケーラブルなオリジンを使用してコンテンツを保存して配信する – ビューワーが使用しているすべてのデバイスに必要な形式でコンテンツが MediaLive でエンコードされている場合は、AWS Elemental MediaStore などの高スケーラブルなオリジンを使用してコンテンツを配信します。CloudFront を使用して MediaStore コンテナに保存されているコンテンツを配信する詳細な手順については、「AWS Elemental MediaStore をオリジンとして使用してビデオを配信する」を参照してください。

これらのオプションのいずれかを使用してオリジンを設定したら、CloudFront を使用して、ビューワーにライブストリーミングビデオを配信できます。

> **Tip**
>
> 可用性の高いリアルタイムの視聴エクスペリエンスを実現するサービスを自動的にデプロイする AWS ソリューションについて、情報を入手することができます。このソリューションを自動的にデプロイする手順については、「ライブストリーミングの自動デプロイ」を参照してください。

トピック

- AWS Elemental MediaStore をオリジンとして使用してビデオを配信する
- AWS Elemental MediaPackage でフォーマットされたライブ動画を配信する

AWS Elemental MediaStore をオリジンとして使用してビデオを配信する

AWS Elemental MediaStore コンテナにビデオを保存している場合は、CloudFront ディストリビューションを作成してコンテンツを配信できます。

開始するには、CloudFront に MediaStore コンテナへのアクセスを許可します。次に、CloudFront ディストリビューションを作成し、MediaStore で使用するように設定します。

AWS Elemental MediaStore コンテナからコンテンツを提供するには

1. 「Amazon CloudFront に AWS Elemental MediaStore コンテナへのアクセスを許可する」の手順に従った後、以下の手順に戻ってディストリビューションを作成します。
2. 次の設定でディストリビューションを作成します。

   a. オリジンドメイン – MediaStore コンテナに割り当てられたデータエンドポイントです。ドロップダウンリストから、ライブビデオ用の MediaStore コンテナを選択します。

   b. オリジンパス – オブジェクトが保存されている MediaStore コンテナのフォルダ構造です。詳細については、「the section called “オリジンのパス”」を参照してください。

   c. カスタムヘッダーを追加 – CloudFront がリクエストをオリジンに転送するときにカスタムヘッダーを追加する場合は、ヘッダーの名前と値を追加します。

   d. ビューワープロトコルポリシー — [HTTP を HTTPS にリダイレクト] を選択します。詳細については、「the section called “ビューワープロトコルポリシー”」を参照してください。

e. キャッシュポリシーとオリジンリクエストポリシー

- [キャッシュポリシー] で、[ポリシーを作成] を選択し、キャッシュのニーズとセグメント期間に適したキャッシュポリシーを作成します。ポリシーを作成したら、キャッシュポリシーのリストを更新し、先ほど作成したポリシーを選択します。
- オリジンリクエストポリシーでは、ドロップダウンリストから [CORS-CustomOrigin] (CORS-CustomOrigin)を選択します。

その他の設定については、他の技術的要件またはビジネスのニーズに基づいて特定の値を設定できます。ディストリビューションのすべてのオプションのリストおよびその設定に関する情報については、「the section called “ディストリビューションの設定”」を参照してください。

3. アプリケーション (メディアプレーヤーなど) 内のリンクの場合は、CloudFront を使用して配信する他のオブジェクトと同じ形式でメディアファイルの名前を指定します。

AWS Elemental MediaPackage でフォーマットされたライブ動画を配信する

AWS Elemental MediaPackage を使用してライブストリームをフォーマットした場合は、CloudFront ディストリビューションを作成し、ライブストリームを配信するようにキャッシュ動作を設定できます。以下のプロセスでは、MediaPackage を使用してライブビデオ用のチャネルの作成とエンドポイントの追加が済んでいることを前提としています。

MediaPackage で CloudFront ディストリビューションを手動で作成するには、以下の手順に従います。

ステップ

- ステップ 1: CloudFront ディストリビューションを作成して設定する
- ステップ 2: MediaPackage エンドポイントのドメインにオリジンを追加する
- ステップ 3: すべてのエンドポイントのキャッシュ動作を設定する
- ステップ 4: ヘッダーベースの MediaPackage CDN 認証を有効にする
- ステップ 5: CloudFront を使用してライブストリームチャネルを提供する

ステップ 1: CloudFront ディストリビューションを作成して設定する

以下の手順を実行して、MediaPackage で作成したライブビデオチャネルの CloudFront ディストリビューションを設定します。

ライブビデオチャネルのディストリビューションを作成するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. [ディストリビューションの作成] を選択します。
3. 以下のようなディストリビューション用の設定を選択します。

   オリジンドメイン

   MediaPackage ライブビデオチャネルとエンドポイントがあるオリジン。テキストフィールドを選択し、ドロップダウンリストから、ライブビデオ用の MediaPackage オリジンドメインを選択します。1 つのドメインを複数のオリジンエンドポイントにマップできます。

   別の AWS アカウントを使用してオリジンドメインを作成した場合は、オリジンの URL 値をフィールドに入力します。オリジンは HTTPS URL であることが必要です。

   例えば、`https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8` などの HLS エンドポイントの場合、オリジンドメインは `3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com` です。

   詳細については、「the section called “オリジンドメイン”」を参照してください。

   オリジンのパス

   コンテンツが配信される MediaPackage エンドポイントへのパス。

   [オリジンパス] フィールドは自動入力されません。正しいオリジンパスを手動で入力する必要があります。

   オリジンパスの仕組みの詳細については、the section called “オリジンのパス” を参照してください。

> **Important**
>
> CloudFront ディストリビューションのどこかにルーティングするには、ワイルドカードパス * が必要です。明示的なパスと一致しないリクエストが実際のオリジンにルーティングされないようにするには、そのワイルドカードパスに「ダミー」オリジンを作成します。

Example : 「ダミー」オリジンの作成

次の例で、エンドポイント abc123 および def456 は「実際の」オリジンにルーティングしますが、他のエンドポイントのビデオコンテンツのリクエストは、適切なサブドメインがないと mediapackage.us-west-2.amazonaws.com にルーティングされ、HTTP 404 エラーになります。

MediaPackage エンドポイント:

```
https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/
index.m3u8
https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/def456/
index.m3u8
```

CloudFront オリジン A:

```
Domain: 3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com
Path: None
```

CloudFront オリジン B:

```
Domain: mediapackage.us-west-2.amazonaws.com
Path: None
```

CloudFront キャッシュビヘイビア:

```
1. Path: /out/v1/abc123/* forward to Origin A
2. Path: /out/v1/def456/* forward to Origin A
3. Path: * forward to Origin B
```

その他のディストリビューション設定には、他の技術的要件またはビジネスニーズに基づいて値を指定します。ディストリビューションのすべてのオプションのリストおよびその設定に関する情報については、「the section called “ディストリビューションの設定”」を参照してください。

他のディストリビューションの設定を選択したら、[Create Distribution] (ディストリビューションの作成) を選択します。

4. 作成したディストリビューションを選択し、[Behaviors] (動作) タブを選択します。
5. デフォルトのキャッシュ動作を選択し、[Edit] (編集) をクリックします。オリジンに対して選択したチャネルの正しいキャッシュ動作設定を指定します。後で他のオリジンを 1 つ以上追加し、それらのキャッシュ動作設定を編集します。
6. CloudFront ディストリビューションページに移動します。
7. ディストリビューションの [Last modified] (最終更新日時) 列の値が [Deploying] (デプロイ中) から日付と時刻に変わって、CloudFront によってディストリビューションが作成されたことが示されるまで待ちます。

ステップ 2: MediaPackage エンドポイントのドメインにオリジンを追加する

この手順を繰り返して、各 MediaPackage チャネルエンドポイントをディストリビューションに追加します。ただし、「ダミー」オリジンを作成する必要があることに注意してください。

オリジンとして他のエンドポイントを追加するには

1. CloudFront コンソールで、チャンネル用に作成したディストリビューションを選択します。
2. [Origins] (オリジン)、[Create origin] (オリジンの作成) の順に選択します。
3. [Origin domain] (オリジンドメイン)では、ドロップダウンリストからチャネルの MediaPackage エンドポイントを選択します。
4. その他の設定には、他の技術的要件またはビジネスニーズに基づいて値を指定します。詳細については、「the section called “オリジンの設定”」を参照してください。
5. [Create Origin] (オリジンの作成) を選択します。

ステップ 3: すべてのエンドポイントのキャッシュ動作を設定する

エンドポイントごとに、キャッシュ動作を設定して、リクエストを正しくルーティングするパスパターンを追加する必要があります。指定するパスパターンは、配信するビデオの形式によって異なり

ます。次の手順には、Apple HLS、CMAF、DASH、および Microsoft スムーズストリーミング形式で使用するパスパターン情報が含まれています。

通常、エンドポイントごとに 2 つのキャッシュ動作を設定します。

- ファイルのインデックスになる親マニフェスト
- セグメント (ビデオコンテンツのファイル)

エンドポイントのキャッシュ動作を作成するには

1. CloudFront コンソールで、チャンネル用に作成したディストリビューションを選択します。
2. [Behaviors] (動作)、[Create Behavior] (動作の作成) の順に選択します。
3. [パスパターン] で、特定の MediaPackage `OriginEndpoint` GUID をパスプレフィックスとして使用します。

   パスパターン

   `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8` などの HLS エンドポイントの場合は、以下の 2 つのキャッシュビヘイビアを作成します。

   - 親および子マニフェストの場合は、`/out/v1/abc123/*.m3u8` を使用します。
   - コンテンツセグメントの場合は、`/out/v1/abc123/*.ts` を使用します。

   `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8` などの CMAF エンドポイントの場合は、以下の 2 つのキャッシュビヘイビアを作成します。

   - 親および子マニフェストの場合は、`/out/v1/abc123/*.m3u8` を使用します。
   - コンテンツセグメントの場合は、`/out/v1/abc123/*.mp4` を使用します。

   `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.mpd` などの DASH エンドポイントの場合は、以下の 2 つのキャッシュビヘイビアを作成します。

   - 親マニフェストの場合は、`/out/v1/abc123/*.mpd` を使用します。
   - コンテンツセグメントの場合は、`/out/v1/abc123/*.mp4` を使用します。

   `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.ism` などの Microsoft Smooth Streaming エンドポイントの場合は、

マニフェストのみが使用されるため、1 つのキャッシュビヘイビア (out/v1/abc123/index.ism/*) のみを作成します。

4. キャッシュ動作ごとに、以下の設定の値を指定します。

ビューワープロトコルポリシー

[Redirect HTTP to HTTPS (HTTP から HTTPS へのリダイレクト)] を選択します。

キャッシュポリシーとオリジンリクエストポリシー

キャッシュポリシーでは、[Create policy] (ポリシーの作成) を選択します。新しいキャッシュポリシーでは、次の設定を指定します。

最小 TTL

古いコンテンツを提供しないように 5 秒以下に設定します。

クエリ文字列

クエリ文字列([Cache key settings] (キャッシュキー設定)) で、[Include specified query strings] (指定したクエリ文字列を含める) を選択します。[Allow] (許可) では、次の値を入力し、[Add item] (アイテムを追加) を選択します。

- CloudFront によってキャッシュの基準として使用されるクエリ文字列パラメータとして m を追加します。MediaPackage レスポンスには、エンドポイントの変更時間を取得するためのタグ ?m=### が常に含まれます。コンテンツがすでにキャッシュされていて、このタグに別の値が設定されている場合、CloudFront はキャッシュされたバージョンを配信する代わりに新しいマニフェストをリクエストします。
- MediaPackage でタイムシフト表示機能を使用している場合は、マニフェストリクエスト (start、end、*.m3u8) のキャッシュ動作の追加のクエリ文字列パラメータとして *.mpd と index.ism/* を指定します。これにより、マニフェストリクエストで指定した期間に固有のコンテンツが配信されます。コンテンツのタイムシフト表示および形式の開始/終了リクエストパラメータの詳細については、「AWS Elemental MediaPackage ユーザーガイド」の「タイムシフト表示」を参照してください。
- MediaPackage でマニフェストフィルタリング機能を使用している場合は、マニフェストリクエスト (*.m3u8、*.mpd、index.ism/*) のキャッシュ動作の追加のクエリ文字列パラメータとして aws.manifestfilter を指定します。これにより、aws.manifestfilter クエリ文字列を MediaPackage オリジンに転送するようにディストリビューションが設定されます。これは、マニフェストフィルタリング機能を有効にするために必要です。詳細については、「AWS Elemental MediaPackage ユーザーガイド」の「マニフェストでのフィルタリング」を参照してください。

- 低レイテンシー HLS (LL-HLS) を使用している場合は、マニフェストリクエスト (`*.m3u8`) のキャッシュビヘイビアで使用するキャッシュポリシーの追加クエリ文字列パラメータとして `_HLS_msn` および `_HLS_part` を指定します。これにより、`_HLS_msn` および `_HLS_part` クエリ文字列を MediaPackage オリジンに転送するようにディストリビューションが設定されます。これは、LL-HLS のプレイリストリクエストのブロック機能を有効にするために必要です。

5. [Create] (作成) を選択します。
6. キャッシュポリシーを作成したら、キャッシュ動作の作成ワークフローに戻ります。キャッシュポリシーのリストを更新し、先ほど作成したポリシーを選択します。
7. [Create behavior] (動作の作成) を選択します。
8. エンドポイントが Microsoft Smooth Streaming エンドポイント以外の場合、次の手順を繰り返して 2 つ目のキャッシュ動作を作成します。

ステップ 4: ヘッダーベースの MediaPackage CDN 認証を有効にする

MediaPackage エンドポイントと CloudFront ディストリビューションの間で、ヘッダーベースの MediaPackage CDN 認証を有効にすることをお勧めします。詳細については、「AWS Elemental MediaPackage ユーザーガイド」の「MediaPackage で CDN 認証を有効にする」を参照してください。

ステップ 5: CloudFront を使用してライブストリームチャネルを提供する

ディストリビューションの作成、オリジンの追加、キャッシュビヘイビアの作成、ヘッダーベースの CDN 認証の有効化が済んだら、CloudFront を使用してライブストリームチャネルを配信できます。CloudFront は、キャッシュ動作用に定義した設定に基づいて、ビューワーからのリクエストを正しい MediaPackage エンドポイントにルーティングします。

アプリケーション (メディアプレーヤーなど) 内のリンクの場合は、CloudFront URL の標準形式でメディアファイルの URL を指定します。詳細については、「the section called “ファイルの URL をカスタマイズする”」を参照してください。

関数を使用してエッジでカスタマイズする

Amazon CloudFront では、独自のコードを記述して、CloudFront ディストリビューションプロセスが HTTP リクエストとレスポンスを処理する方法をカスタマイズできます。このコードは、レイテンシーを最小限に抑えるためにビューワー (ユーザー) の近くで実行するため、サーバーやその他のインフラストラクチャを管理する必要はありません。CloudFront を通過するリクエストとレスポンスの操作、基本認証と承認の実行、エッジでの HTTP レスポンスの生成などのコードを記述できます。

CloudFront ディストリビューションに記述してアタッチするコードは、エッジ関数と呼ばれます。CloudFront でエッジ関数の記述と管理をする方法は 2 つあります。

CloudFront Functions

大規模でレイテンシーの影響を受けやすい CDN カスタマイズのための軽量な関数を JavaScript で記述できます。CloudFront Functions の runtime 環境は、起動時間が 1 ミリ秒未満、毎秒数百万のリクエストを処理するようにすぐにスケールでき、高い安全性を誇ります。CloudFront Functions は、CloudFront のネイティブ関数です。つまり、CloudFront 内でコードを完全に構築、テスト、デプロイできます。

Lambda@Edge

Lambda@Edge は、複雑な関数および完全なアプリケーションロジックをビューワーに近い場所で実行するための強力で柔軟なコンピューティングを提供する、安全性に優れた AWS Lambda の拡張機能です。Lambda@Edge 関数は、Node.js または Python runtime 環境で実行されます。これらの関数の発行先は単一の AWS リージョンですが、CloudFront ディストリビューションに関数を関連付けると、Lambda@Edge がコードを世界中に自動的にレプリケートします。

CloudFront で AWS WAF を実行する場合、AWS WAF の挿入されたヘッダーを CloudFront Functions と Lambda@Edge の両方で使用できます。これはビューワーとオリジンのリクエストとレスポンスで機能します。

トピック

- CloudFront Functions と Lambda@Edge の違い
- CloudFront Functions を使用してエッジでカスタマイズする
- Lambda@Edge を使用してエッジでカスタマイズする
- エッジ関数に対する制限

CloudFront Functions と Lambda@Edge の違い

CloudFront Functions と Lambda@Edge は、どちらでもCloudFront イベントに応答してコードを実行できます。

CloudFront Functions は、以下のユースケースにおける軽量で実行時間の短い関数に最適です。

- キャッシュキーの正規化 – HTTP リクエスト属性 (ヘッダー、クエリ文字列、Cookie、さらには URL パス) を変換して、最適なキャッシュキーを作成します。これにより、キャッシュのヒット率を向上させることができます。
- ヘッダー操作 – リクエストまたはレスポンスに対する HTTP ヘッダーの挿入、変更、または削除を行います。たとえば、すべてのリクエストに `True-Client-IP` ヘッダーを追加できます。
- URL リダイレクトまたは書き換え – リクエスト内の情報に基づいてビューワーを他のページにリダイレクトしたり、あるパスから別のパスにすべてのリクエストを書き換えたりします。
- リクエストの承認 – 承認ヘッダーや他のリクエストメタデータを調べることで、JSON ウェブトークン (JWT) などのハッシュ化された承認トークンを検証します。

CloudFront Functions の使用を開始するには、「CloudFront Functions を使用してエッジでカスタマイズする」を参照してください。

Lambda@Edge は、以下のユースケースに最適です。

- 完了までに数ミリ秒以上かかる関数。
- 調整可能な CPU またはメモリを必要とする機能。
- サードパーティライブラリに依存する関数 (他の AWS のサービスとの統合のため、AWS SDK を含む)。
- 外部サービスを使用して処理するために、ネットワークアクセスを必要とする関数。
- ファイルシステムへのアクセスまたは HTTP リクエストの本文へのアクセスを必要とする関数。

Lambda@Edge の使用を開始するには、「Lambda@Edge を使用してエッジでカスタマイズする」を参照してください。

次の表を使用して CloudFront Functions と Lambda@Edge の違いを理解すると、ユースケースのオプションを選択しやすくなります。

	CloudFront Functions	Lambda@Edge
プログラミング言語	JavaScript (ECMAScript 5.1 準拠)	Node.js と Python
イベントソース	• ビューワーリクエスト • ビューワーレスポンス	• ビューワーリクエスト • ビューワーレスポンス • オリジンリクエスト • オリジンレスポンス
Amazon CloudFront KeyValueStore をサポートする	はい CloudFront KeyValueStore は JavaScript ランタイム 2.0 のみをサポートします	いいえ
Scale	リクエスト数: 毎秒 10,000,000 件以上	リクエスト数: 1 リージョンあたり毎秒 10,000 件まで
関数の持続時間	サブミリ秒	最大 5 秒 (ビューワーリクエスト、ビューワーレスポンス) 最大 30 秒 (オリジンリクエスト、オリジンレスポンス)
最大メモリ 詳細については、「Lambda クォータ」を参照してください。	2 MB	128 MB – 10,240 MB (10 GB)
関数コードと含まれるライブラリの最大サイズ	10 KB	1 MB (ビューワーリクエスト、ビューワーレスポンス)

	CloudFront Functions	Lambda@Edge
		50 MB (オリジンリクエスト、オリジンレスポンス)
ネットワークアクセス	いいえ	はい
ファイルシステムへのアクセス	いいえ	はい
リクエスト本文へのアクセス	いいえ	はい
位置情報とデバイスデータへのアクセス	はい	いいえ (ビューワーリクエスト、ビューワーレスポンス) はい (オリジンリクエスト、オリジンレスポンス)
CloudFront でビルドとテストをすべて実施可能	はい	いいえ
関数のログとメトリクス	はい	はい
料金表	無料利用枠あり。リクエストごとに課金。	無料利用枠なし。リクエストと機能期間ごとに課金。

CloudFront Functions を使用してエッジでカスタマイズする

CloudFront Functions を使用すると、JavaScript で軽量な関数を記述し、レイテンシーの影響を受けやすい CDN カスタマイズを大規模に実行できます。関数を使用して、CloudFront を通過するリクエストとレスポンスの操作、基本認証と承認の実行、エッジでの HTTP レスポンスの生成などを行うことができます。CloudFront Functions の runtime 環境は、起動時間が 1 ミリ秒未満、毎秒数百万のリクエストを処理するようにすぐにスケールでき、高い安全性を誇ります。CloudFront Functions は、CloudFront のネイティブ関数です。つまり、CloudFront 内でコードを完全に構築、テスト、デプロイできます。

CloudFront 関数を CloudFront ディストリビューションに関連付けると、CloudFront が CloudFront エッジロケーションでリクエストとレスポンスをインターセプトし、この関数に送ります。以下のイベントが発生したら、CloudFront Functions を呼び出すことができます。

- CloudFront がビューワーからリクエストを受信したとき (ビューワーリクエスト)
- CloudFront がビューワーにレスポンスを返す前 (ビューワーレスポンス)

CloudFront Functions の詳細については、以下のトピックを参照してください。

トピック

チュートリアル: CloudFront Functions でシンプルな関数を作成する

このチュートリアルでは、CloudFront Functions の使用を開始する方法を示します。ビューワーを別の URL にリダイレクトし、さらにカスタムレスポンスヘッダーを返すシンプルな関数を作成できます。

目次

前提条件

CloudFront Functions の使用には、CloudFront ディストリビューションが必要です。アカウントをお持ちでない場合は、「基本的な CloudFront ディストリビューションの開始方法」を参照してください。

関数を作成する

CloudFront コンソールを使用して、ビューワーを別の URL にリダイレクトし、さらにカスタムレスポンスヘッダーを返すシンプルな関数を作成できます。

CloudFront 関数を作成するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[関数]、[関数を作成] の順に選択します。
3. [関数を作成] ページの [名前] に関数名 (*MyFunctionName* など) を入力します。
4. (オプション) [説明] に、関数の説明 (**Simple test function** など) を入力します。
5. [ランタイム] は、デフォルトで選択されている JavaScript バージョンのままにします。
6. [Create function (関数の作成)] を選択します。
7. 次の関数コードをコピーします。この関数コードは、ビューワーを別の URL にリダイレクトするほか、カスタムレスポンスヘッダーも返します。

```
function handler(event) {
    // NOTE: This example function is for a viewer request event trigger.
    // Choose viewer request for event trigger when you associate this function
 with a distribution.
    var response = {
        statusCode: 302,
        statusDescription: 'Found',
        headers: {
            'cloudfront-functions': { value: 'generated-by-CloudFront-Functions' },
            'location': { value: 'https://aws.amazon.com/cloudfront/' }
        }
    };
    return response;
}
```

8. [関数コード] で、コードをコードエディタに貼り付け、デフォルトコードを置き換えます。

9. [Save changes] (変更の保存) をクリックします。
10. (オプション) 関数は発行前にテストすることもできます。このチュートリアルでは、関数のテスト方法は説明していません。詳細については、「関数をテストする」を参照してください。
11. [発行] タブを選択し、[関数を発行] を選択します。関数は、CloudFront ディストリビューションと関連付ける前に発行する必要があります。
12. 次に、関数をディストリビューションまたはキャッシュビヘイビアと関連付けることができます。*[MyFunctionName]* ページで、[発行] タブを選択します。

> **Warning**
>
> 次の手順では、テストに使用するディストリビューションまたはキャッシュビヘイビアを選択します。このテスト関数は、本番環境で使用するディストリビューションやキャッシュビヘイビアには関連付けないでください。

13. [Add association] を選択します。
14. [関連付け] ダイアログボックスで、ディストリビューションまたはキャッシュビヘイビアを選択します。[イベントタイプ] は、デフォルト値のままにします。
15. [Add association] を選択します。

    関連付けられているディストリビューションは、[関連付けられているディストリビューション] テーブルに表示されます。
16. 関連付けられたディストリビューションのデプロイが完了するまで数分待ちます。ディストリビューションのステータスを確認するには、[関連づけられているディストリビューション] テーブルでディストリビューションを選択し、[ディストリビューションを表示] を選択します。

    ディストリビューションのステータスが [Deployed] になったら、関数の動作を確認できます。

関数を検証する

関数をデプロイしたら、ディストリビューションで機能することを検証できます。

関数を検証するには

1. ウェブブラウザで、ディストリビューションのドメイン名 (`https://d111111abcdef8.cloudfront.net` など) に移動します。

関数はブラウザにリダイレクトを返すため、ブラウザは自動的に https://aws.amazon.com/cloudfront/ に移動します。

2. コマンドラインウィンドウで、curl などのツールを使用してディストリビューションのドメイン名にリクエストを送信できます。

```
curl -v https://d111111abcdef8.cloudfront.net/
```

レスポンスに、リダイレクトレスポンス (302 Found) と、関数が追加したカスタムレスポンスヘッダーが表示されます。レスポンスは次の例のようになります。

Example

```
curl -v https://d111111abcdef8.cloudfront.net/
> GET / HTTP/1.1
> Host: d111111abcdef8.cloudfront.net
> User-Agent: curl/7.64.1
> Accept: */*
>
< HTTP/1.1 302 Found
< Server: CloudFront
< Date: Tue, 16 Mar 2021 18:50:48 GMT
< Content-Length: 0
< Connection: keep-alive
< Location: https://aws.amazon.com/cloudfront/
< Cloudfront-Functions: generated-by-CloudFront-Functions
< X-Cache: FunctionGeneratedResponse from cloudfront
< Via: 1.1 3035b31bddaf14eded329f8d22cf188c.cloudfront.net (CloudFront)
< X-Amz-Cf-Pop: PHX50-C2
< X-Amz-Cf-Id: ULZdIz6j43uGBlXyob_JctF9x7CCbwpNniiMlmNbmwzH1YWP9FsEHg==
```

チュートリアル: キー値を含む関数を作成する

このチュートリアルでは、CloudFront Function にキー値を含める方法を説明します。キー値はキーと値のペアの一部です。関数コードには (キーと値のペアの) 名前を含めます。関数を実行すると、CloudFront は名前を値に置き換えます。

キーと値のペアはキー値ストアに保存される変数です。関数で (ハードコードされた値の代わりに) キーを使用すると、関数の柔軟性が高まります。キーの値は、コードの変更をデプロイしなくても変

更できます。キーと値のペアによって関数のサイズを小さくすることもできます。詳細については、「???」を参照してください。

目次

前提条件

CloudFront Functions の関数とキー値ストアを初めて使用する場合は、「the section called “チュートリアル: シンプルな CloudFront 関数を作成する”」のチュートリアルに従うことをお勧めします。

そのチュートリアルを完了したら、このチュートリアルに従って、作成した関数を拡張できます。このチュートリアルでは、最初にキー値ストアを作成することをお勧めします。

キー値ストアを作成する

まず、関数で使用するキー値ストアを作成します。

キー値ストアを作成するには

1. 関数に含めるキーと値のペアを計画します。キーの名前を書き留めます。1 つの関数で使用するキーと値のペアは、すべてが 1 つのキー値ストア内に存在する必要があります。
2. 作業の順序を決めます。次の 2 通りの方法があります。
   - キー値ストアを作成してキーと値のペアを追加します。次に、関数を作成 (または変更) し、キー名を組み込みます。
   - または、関数を作成 (または変更) し、使用したいキー名を組み込みます。次にキー値ストアを作成し、キーと値のペアを追加します。
3. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
4. ナビゲーションペインで、[関数]、[KeyValueStores] タブの順に選択します。
5. [KeyValueStore を作成] を選択し、以下のフィールドに入力します。

- ストアの名前と、オプションの説明を入力します。
- [S3 URI] は空白のままにします。このチュートリアルでは、キーと値のペアを手動で入力します。

6. [Create] (作成) を選択します。新しいキー値ストアの詳細ページが表示されます。このページには、現在空欄の [キーと値のペア] セクションがあります。

キー値ストアにキーと値のペアを追加する

次に、先ほど作成したキー値ストアにキーと値のペアのリストを手動で追加します。

キー値ストアにキーと値のペアを追加するには

1. [キーと値のペア] セクションで、[キーと値のペアを追加] を選択します。
2. [ペアを追加] を選択し、キーと値を入力します。チェックマークをオンにして変更を確認します。さらに追加するには、このステップを繰り返します。
3. 完了したら、[変更を保存] を選択してキー値ストアにキーと値のペアを保存します。確認ダイアログボックスで、[完了] を選択します。

これで、キーと値のペアのグループがキー値ストアに含まれました。

キー値ストアを関数に関連付ける

これで、キー値ストアが作成されました。また、キー値ストアのキー名を含む関数を作成または変更しました。これでキー値ストアと関数を関連付けることができます。その関連付けは関数内から作成します。

キー値ストアを関数に関連付けるには

1. ナビゲーションペインで、[関数] を選択します。デフォルトでは、[関数] タブが一番上に表示されます。
2. 関数名を選択し、[関連付けられている KeyValueStore] セクションで、[既存の KeyValueStore を関連付ける] を選択します。
3. キー値ストアを選択し、[KeyValueStore を関連付ける] を選択します。

> **Note**
>
> 各関数に関連付けることができるキー値ストアは 1 つだけです。

関数コードをテストして発行する

キー値ストアを関数に関連付けると、関数コードをテストして発行できます。以下を行う場合を含め、関数コードを変更するたびに必ずテストする必要があります。

- キー値ストアを関数に関連付ける
- 関数およびキー値ストアを変更して、新しいキーと値のペアを含めます。
- キーと値のペアの値を変更する

関数コードをテストして発行するには

1. 関数をテストする方法については、「the section called “関数をテストする”」を参照してください。DEVELOPMENT ステージで関数のテストを選択するようにしてください。
2. 関数を (新規または更新されたキーと値のペアと共に) LIVE 環境で使用する準備が整ったら、関数を発行します。

   発行すると、CloudFront は関数のバージョンを DEVELOPMENT ステージからライブステージにコピーします。関数には新しいコードが含まれ、キー値ストアに関連付けられます。(ライブステージで関連付けを再度実行する必要はありません)。

   関数を発行する方法については、「the section called “関数を発行する”」を参照してください。

関数コードを記述する

CloudFront Functions を使用すると、大規模でレイテンシーの影響を受けやすい CDN カスタマイズのための軽量な関数を JavaScript で記述できます。関数コードを使用して、CloudFront を通過するリクエストとレスポンスの操作、基本認証と承認の実行、エッジでの HTTP レスポンスの生成などを行うことができます。

CloudFront Functions の関数コードを記述する場合は、以下のトピックも参考になります。

トピック

- 関数の目的を決定する

- CloudFront Functions のイベント構造
- CloudFront Functions の JavaScript ランタイムの機能
- キー値ストアのヘルパーメソッド
- CloudFront Functions のコード例

関数の目的を決定する

関数コードを記述する前に、関数の目的を決めます。CloudFront Functions の関数の大半は、次のいずれかを目的としています。

トピック

- ビューワーリクエストイベントタイプの HTTP リクエストの変更
- ビューワーリクエストイベントタイプで HTTP レスポンスを生成する
- ビューワーレスポンスイベントタイプの HTTP レスポンスの変更
- 関連情報

関数の目的にかかわらず、`handler` はあらゆる関数のエントリポイントです。CloudFront から関数に送られる `event` という単一の引数を使います。`event` は、HTTP リクエスト表記の JSON オブジェクトです (関数が HTTP レスポンスを変更する場合は、レスポンス) 。

ビューワーリクエストイベントタイプの HTTP リクエストの変更

関数は、CloudFront がビューワー (クライアント) から受信する HTTP リクエストを変更し、変更されたリクエストを CloudFront に返して処理を続行できます。たとえば、関数コードでキャッシュキーを正規化したり、リクエストヘッダーを変更したりできます。

HTTP リクエストを変更する関数を作成する場合は、必ずビューワーリクエストイベントタイプを選択してください 。すなわち、関数は CloudFront がビューワーからリクエストを受信するたびに、リクエストされたオブジェクトが CloudFront キャッシュにあるかどうかをチェックする前に実行されます。

Example 例

次の擬似コードは、HTTP リクエストを変更する関数の構造を示しています。

```
function handler(event) {
    var request = event.request;
```

```
    // Modify the request object here.

    return request;
}
```

この関数は、変更された `request` オブジェクトを CloudFront に返します。CloudFront は、CloudFront キャッシュでキャッシュヒットがないかチェックし、必要に応じてリクエストをオリジンに送信することで、返されたリクエストの処理を続行します。

ビューワーリクエストイベントタイプで HTTP レスポンスを生成する

関数は、エッジで HTTP レスポンスを生成し、キャッシュされたレスポンスまたは CloudFront によるそれ以降の処理をチェックすることなく、ビューワー (クライアント) に直接返すことができます。たとえば、関数コードは、リクエストを新しい URL にリダイレクトしたり、承認をチェックして、`401` や `403` レスポンスを非承認リクエストに返す場合があります。

HTTP レスポンスを生成する関数を作成する場合は、必ずビューワーリクエストイベントタイプを選択してください 。つまりこの関数は、CloudFront がビューワーからリクエストを受信するたびに、CloudFront がリクエストの処理を行う前に実行されます。

Example 例

次の擬似コードは、HTTP 応答を生成する関数の構造を示しています。

```
function handler(event) {
    var request = event.request;

    var response = ...; // Create the response object here,
                        // using the request properties if needed.

    return response;
}
```

この関数は、CloudFront に `response` オブジェクトを返します。CloudFront はこのオブジェクトを、CloudFront キャッシュをチェックしたり、オリジンにリクエストを送信したりすることなく、すぐにビューワーに戻します。

ビューワーレスポンスイベントタイプの HTTP レスポンスの変更

関数は、レスポンスが CloudFront キャッシュとオリジンのどちらから来たかに関係なく、CloudFront がビューワー (クライアント) に送信する前に HTTP レスポンスを変更できます。例

えば、関数コードでレスポンスヘッダー、ステータスコード、本文のコンテンツを追加または変更する場合があります。

HTTP レスポンスを変更する関数を作成する場合は、必ずビューワーレスポンスイベントタイプを選択してください。つまり関数は、レスポンスが CloudFront キャッシュまたはオリジンのどちらから来たかに関係なく、CloudFront がビューワーにレスポンスを返す前に実行されます。

Example 例

次の擬似コードは、HTTP レスポンスを変更する関数の構造を示しています。

```
function handler(event) {
    var request = event.request;
    var response = event.response;

    // Modify the response object here,
    // using the request properties if needed.

    return response;
}
```

この関数は変更された `response` オブジェクトを CloudFront に返し、CloudFront はこれをすぐにビューワーに返します。

関連情報

CloudFront Functions の操作の詳細については、以下のトピックを参照してください。

- イベントの構造
- JavaScript ランタイムの機能
- コードの例
- エッジ関数に対する制限

CloudFront Functions のイベント構造

CloudFront Functions は、関数を実行するときに `event` オブジェクトを関数コードに入力として渡します。関数をテストするときは、`event` オブジェクトを作成し、関数に渡します。関数をテストするために `event` オブジェクトを作成する場合は、`context` オブジェクト内の `distributionDomainName`、`distributionId`、`requestId` フィールドを省略できます。ヘッ

ダーの名前が小文字であることを確認してください (CloudFront Functions が本番環境で関数に渡す event オブジェクトはすべて該当します)。

次に、このイベントオブジェクトの構造の概要を示します。

```
{
    "version": "1.0",
    "context": {
        <context object>
    },
    "viewer": {
        <viewer object>
    },
    "request": {
        <request object>
    },
    "response": {
        <response object>
    }
}
```

詳細については、次のトピックを参照してください。

トピック

- バージョンフィールド
- コンテキストオブジェクト
- ビューワーオブジェクト
- リクエストオブジェクト
- レスポンスオブジェクト
- ステータスコードと本文
- クエリ文字列、ヘッダー、または Cookie の構造
- レスポンスオブジェクトの例
- イベントオブジェクトの例

バージョンフィールド

version フィールドには、CloudFront Functions イベントオブジェクトのバージョンを指定する文字列が含まれます。現在のバージョンは 1.0 です。

コンテキストオブジェクト

context オブジェクトには、イベントに関するコンテキスト情報が含まれます。次のフィールドが含まれています。

distributionDomainName

イベントに関連付けられたディストリビューションの CloudFront ドメイン名 (例: d111111abcdef8.cloudfront.net)。

distributionId

イベントに関連付けられたディストリビューションの ID (例: EDFDVBD6EXAMPLE) 。

eventType

イベントタイプ (viewer-request または viewer-response)。

requestId

CloudFront リクエスト (およびそれに関連付けられたレスポンス) を一意に識別する文字列。

ビューワーオブジェクト

viewer オブジェクトには、リクエストを送信したビューワー (クライアント) の IP アドレスを値とする ip フィールドが含まれています。ビューワーリクエストが HTTP プロキシまたはロードバランサーを通して来た場合、値はプロキシまたはロードバランサーの IP アドレスです。

リクエストオブジェクト

request オブジェクトには、ビューワーから CloudFront への HTTP リクエスト表記が含まれています。関数に渡される event オブジェクトで、request オブジェクトは CloudFront がビューワーから受信した実際のリクエストを表しています。

関数コードが CloudFront に request オブジェクトを返す場合は、これと同じ構造を使用する必要があります。

request オブジェクトには、以下のフィールドが含まれています。

method

リクエストの HTTP メソッド。関数コードが request を返す場合、このフィールドは変更できません。これは、request オブジェクト内唯一の読み取り専用フィールドです。

uri

リクエストされたオブジェクトの相対パス。

> **Note**
>
> 関数が uri 値を変更する場合、以下が適用されます。
>
> - 新しい uri 値は、フォワードスラッシュ (/) で始まる必要があります。
> - 関数で uri 値を変更すると、ビューワーがリクエストしているオブジェクトが変更されます。
> - 関数で uri 値を変更しても、リクエストのキャッシュ動作やオリジンリクエストの送信先は変わりません。

querystring

リクエストのクエリ文字列を表すオブジェクト。リクエストにクエリ文字列が含まれていない場合でも、request オブジェクトには空の querystring オブジェクトが含まれています。

querystring オブジェクトには、リクエストのクエリ文字列パラメータ 1 つにつき 1 つのフィールドが含まれます。

headers

リクエストの HTTP ヘッダーを表すオブジェクト。リクエストに Cookie ヘッダーが含まれている場合、それらのヘッダーは headers オブジェクトの一部ではありません。Cookieは cookies オブジェクトで個別に表示されます。

headers オブジェクトには、リクエストのヘッダー 1 つにつき 1 つのフィールドが含まれます。ヘッダー名は、イベントオブジェクトでは小文字に変換されます。また、関数コードで追加する場合、ヘッダー名を小文字にする必要があります。CloudFront Functions がイベントオブジェクトを HTTP リクエストに変換し直すと、ヘッダー名の各単語の最初の文字が大文字になります。各単語はハイフン (-) で区切られます。例えば、関数コードが example-header-name という名前のヘッダーを追加した場合、CloudFront がこれを HTTP リクエストで Example-Header-Name に変換します。

cookies

リクエスト (Cookie ヘッダー) の Cookie を表すオブジェクト。

cookies オブジェクトには、リクエストの Cookie 1 つにつき 1 つのフィールドが含まれます。

クエリ文字列、ヘッダーおよび Cookie の構造の詳細については、「クエリ文字列、ヘッダー、または Cookie の構造」を参照してください。

event オブジェクトの例については、「イベントオブジェクトの例」を参照してください。

レスポンスオブジェクト

response オブジェクトには、CloudFront Front からビューワーへの HTTP レスポンス表記が含まれています。関数に渡される event オブジェクトでは、response オブジェクトはビューワーリクエストに対する CloudFront の実際の応答を表します。

関数コードが response オブジェクトを返す場合は、これと同じ構造を使用する必要があります。

response オブジェクトには、以下のフィールドが含まれています。

statusCode

レスポンスの HTTP ステータスコード。この値は文字列ではなく整数です。

関数は statusCode を生成または変更できます。

statusDescription

レスポンスの HTTP ステータスの説明。関数コードがレスポンスを生成する場合、このフィールドはオプションです。

headers

レスポンスの HTTP ヘッダーを表すオブジェクト。レスポンスに Set-Cookie ヘッダーが含まれている場合、それらのヘッダーは headers オブジェクトの一部ではありません。Cookieは cookies オブジェクトで個別に表示されます。

headers オブジェクトには、レスポンス内のヘッダー 1 つにつき 1 つのフィールドが含まれます。ヘッダー名は、イベントオブジェクトでは小文字に変換されます。また、関数コードで追加する場合、ヘッダー名を小文字にする必要があります。CloudFront Functions がイベントオブジェクトを HTTP レスポンスに変換し直すと、ヘッダー名の各単語の最初の文字が大文字になります。各単語はハイフン (-) で区切られます。例えば、関数コードが example-header-name という名前のヘッダーを追加した場合、CloudFront がこれを HTTP レスポンスの Example-Header-Name に変換します。

cookies

レスポンス (Set-Cookie ヘッダー) で Cookie を表すオブジェクト。

cookies オブジェクトには、レスポンスの Cookie 1 つにつき 1 つのフィールドが含まれます。

body

body フィールドの追加はオプションで、関数で指定しない限り response オブジェクトには表示されません。関数は、CloudFront キャッシュまたはオリジンによって返された元の本文にアクセスできません。ビューワーレスポンス関数で body フィールドを指定しない場合、CloudFront キャッシュまたはオリジンから返された元の本文がビューワーに返されます。

CloudFront がビューワーにカスタム本文を返すようにするには、data フィールドに本文コンテンツを指定し、encoding フィールドの本文エンコーディングを指定します。エンコーディングは、プレーンテキスト ("encoding": "text") または Base64 でエンコードされたコンテンツ ("encoding": "base64") として指定できます。

ショートカットとして、本文の内容を body フィールド ("body": "<specify the body content here>") で直接指定することもできます。これを行うときは、data および encoding フィールドを省略してください。この場合、CloudFront は、本文をプレーンテキストとして扱います。

encoding

body コンテンツ (data フィールド) のエンコーディング。有効なエンコードは text と base64 のみです。

encoding を base64 と指定したが本文が有効な base64 でない場合、CloudFront はエラーを返します。

data

body コンテンツ。

変更されたステータスコードと本文の内容の詳細については、ステータスコードと本文 を参照してください。

ヘッダーと Cookie の構造の詳細については、「クエリ文字列、ヘッダー、または Cookie の構造」を参照してください。

response オブジェクトの例については、「レスポンスオブジェクトの例」を参照してください。

ステータスコードと本文

CloudFront Functions を使用して、ビューワーのレスポンスステータスコードを更新したり、レスポンス本文すべてを新しく置き換えたり、レスポンス本文を削除したりできます。CloudFront キャッシュまたはオリジンからのレスポンスを評価した後でビューワーのレスポンスを更新する一般的なシナリオには、次のようなものがあります。

- ステータスを変更して HTTP 200 ステータスコードを設定し、ビューワーに返す静的な本文コンテンツを作成する。
- HTTP 301 または 302 ステータスコードを設定して、ユーザーを別のウェブサイトにリダイレクトする。
- ビューワーレスポンスの本文を配信するか削除するかを決定します。

> **Note**
>
> オリジンが 400 以上の HTTP エラーを返した場合、CloudFront Functions は実行されません。詳細については、「すべてのエッジ機能に対する制限」を参照してください。

HTTP レスポンスを使用する場合、CloudFront Functions は、レスポンス本文にアクセスできません。必要な値に設定することで本文コンテンツを置き換えたり、値を空に設定することで本文を削除したりできます。関数内の本文フィールドを更新しない場合は、CloudFront キャッシュまたはオリジンによって返された元の本文がビューワーに返されます。

> **Tip**
>
> CloudFront Functions を使用して本文を置き換える場合は、`content-encoding`、`content-type`、`content-length` などの対応するヘッダーを新しい本文のコンテンツに合わせてください。
> たとえば、CloudFront オリジンまたはキャッシュが `content-encoding: gzip` を返したが、ビューワーレスポンス関数が本文をプレーンテキストに設定した場合、関数は `content-encoding` と `content-type` ヘッダーもそれに応じて変更する必要があります。

CloudFront Functions が 400 以上の HTTP エラーを返すように設定されている場合、ビューワーには同じステータスコードに対して指定したカスタムエラーページは表示されません。

クエリ文字列、ヘッダー、または Cookie の構造

クエリ文字列、ヘッダー、Cookie は同じ構造を共有します。クエリ文字列は、リクエストに表示される場合があります。ヘッダーは、リクエストとレスポンスに表示されます。Cookie は、リクエストとレスポンスに表示されます。

クエリ文字列、ヘッダーおよび Cookie はすべて、親 `querystring`、`headers`、`cookies`オブジェクトで一意のフィールドです。フィールド名は、クエリ文字列、ヘッダー、または Cookie の名前です。各フィールドには、クエリ文字列、ヘッダー、Cookie の値を持つ `value` プロパティが含まれます。

目次

クエリ文字列値またはクエリ文字列オブジェクト

関数は、クエリ文字列オブジェクトに加えてクエリ文字列値を返すことができます。クエリ文字列値を使用して、クエリ文字列パラメータを任意のカスタム順序で配置できます。

Example 例

関数コードでクエリ文字列を変更するには、次のようなコードを使用します。

```
var request = event.request;
request.querystring =
 'ID=42&Exp=1619740800&TTL=1440&NoValue=&querymv=val1&querymv=val2,val3';
```

ヘッダーに関する特別な考慮事項

ヘッダーのみの場合、ヘッダー名がイベントオブジェクトで小文字に変換されます。また、関数コードで追加する場合は、ヘッダー名を小文字にする必要があります。CloudFront Functions がイベントオブジェクトを HTTP リクエストまたはレスポンスに変換し直すと、ヘッダー名の各単語の最初の文字が大文字になります。各単語はハイフン (-) で区切られます。例えば、関数コードが `example-header-name` という名前のヘッダーを追加した場合、CloudFront がこれを HTTP リクエストまたはレスポンスの `Example-Header-Name` に変換します。

Example 例

HTTP リクエストでの次の Host ヘッダーについて考えてみます。

```
Host: video.example.com
```

このヘッダーは、request オブジェクトで次のように表されます。

```
"headers": {
    "host": {
        "value": "video.example.com"
    }
}
```

関数コードの Host ヘッダーにアクセスするには、次のようなコードを使用します。

```
var request = event.request;
var host = request.headers.host.value;
```

関数コードでヘッダーを追加または変更するには、次のようなコードを使用します (このコードは、X-Custom-Header 値で example value という名前のヘッダーを追加します)。

```
var request = event.request;
request.headers['x-custom-header'] = {value: 'example value'};
```

重複するクエリ文字列、ヘッダー、Cookie (**multiValue** 配列)

HTTP リクエストまたはレスポンスには、同じ名前のクエリ文字列、ヘッダー、Cookie が含まれることがあります。この場合、重複するクエリ文字列、ヘッダー、Cookie は request または response オブジェクトの 1 つのフィールドに折りたたまれていますが、このフィールドには multiValue という名前の追加のプロパティが含まれます。multiValue プロパティには、重複するクエリ文字列、ヘッダー、Cookie の各値を含む配列が含まれます。

Example 例

以下の Accept ヘッダーを持つ HTTP リクエストについて考えてみます。

```
Accept: application/json
```

```
Accept: application/xml
Accept: text/html
```

これらのヘッダーは、request オブジェクトで次のように表されます。

```
"headers": {
    "accept": {
        "value": "application/json",
        "multiValue": [
            {
                "value": "application/json"
            },
            {
                "value": "application/xml"
            },
            {
                "value": "text/html"
            }
        ]
    }
}
```

> **Note**
>
> 最初のヘッダー値 (この場合は application/json) は、value プロパティと multiValue プロパティの両方で繰り返されています。これにより、multiValue 配列をループしてすべての値にアクセスできます。

関数コードで変更するクエリ文字列、ヘッダー、または Cookie に multiValue 配列が含まれている場合、CloudFront Functions は以下のルールを使用して変更を適用します。

1. multiValue 配列が存在し、変更がある場合は、その変更が適用されます。value プロパティの最初の要素は無視されます。
2. それ以外の場合は、value プロパティへの変更が適用され、それ以降の値 (存在する場合) は変更されません。

この multiValue プロパティは、前の例に示すように、HTTP リクエストまたはレスポンスに同じ名前の重複するクエリ文字列、ヘッダー、Cookie のいずれかが含まれている場合にのみ使用されま

す。ただし、1 つのクエリ文字列、ヘッダー、または Cookie に複数の値がある場合、multiValue プロパティは使用されません。

Example 例

3 つの値を含む 1 つの Accept ヘッダーを持つリクエストについて考えてみます。

```
Accept: application/json, application/xml, text/html
```

このヘッダーは、request オブジェクトで次のように表されます。

```
"headers": {
    "accept": {
        "value": "application/json, application/xml, text/html"
    }
}
```

Cookie 属性

HTTP レスポンスの Set-Cookie ヘッダーでは、ヘッダーに Cookie の名前と値のペア、および必要に応じてセミコロンで区切られた属性のセットが含まれます。

Example 例

```
Set-Cookie: cookie1=val1; Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
 07:28:00 GMT
```

response オブジェクトでは、これらの属性は Cookie フィールドの attributes プロパティで表されます。たとえば、前の Set-Cookie ヘッダーは次のように表されます。

```
"cookie1": {
    "value": "val1",
    "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
 07:28:00 GMT"
}
```

レスポンスオブジェクトの例

次の例は、本文がビューワーレスポンス関数に置き換えられた response オブジェクト (ビューワーレスポンス関数の出力) を示しています。

```
{
  "response": {
    "statusCode": 200,
    "statusDescription": "OK",
    "headers": {
      "date": {
        "value": "Mon, 04 Apr 2021 18:57:56 GMT"
      },
      "server": {
        "value": "gunicorn/19.9.0"
      },
      "access-control-allow-origin": {
        "value": "*"
      },
      "access-control-allow-credentials": {
        "value": "true"
      },
      "content-type": {
        "value": "text/html"
      },
      "content-length": {
        "value": "86"
      }
    },
    "cookies": {
      "ID": {
        "value": "id1234",
        "attributes": "Expires=Wed, 05 Apr 2021 07:28:00 GMT"
      },
      "Cookie1": {
        "value": "val1",
        "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
 07:28:00 GMT",
        "multiValue": [
          {
            "value": "val1",
            "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
 07:28:00 GMT"
          },
          {
            "value": "val2",
            "attributes": "Path=/cat; Domain=example.com; Expires=Wed, 10 Jan 2021
 07:28:00 GMT"
```

```
          }
        ]
      }
    },

    // Adding the body field is optional and it will not be present in the response
 object
    // unless you specify it in your function.
    // Your function does not have access to the original body returned by the
 CloudFront
    // cache or origin.
    // If you don't specify the body field in your viewer response function, the
 original
    // body returned by the CloudFront cache or origin is returned to viewer.

     "body": {
      "encoding": "text",
      "data": "<!DOCTYPE html><html><body><p>Here is your custom content.</p></body></
html>"
    }
  }
}
```

イベントオブジェクトの例

以下は、完全な event オブジェクトの例です。

> **Note**
>
> event オブジェクトは関数への入力です。関数は、event オブジェクト全体ではなく、request または response オブジェクトだけを返します。

```
{
    "version": "1.0",
    "context": {
        "distributionDomainName": "d111111abcdef8.cloudfront.net",
        "distributionId": "EDFDVBD6EXAMPLE",
        "eventType": "viewer-response",
        "requestId": "EXAMPLEntjQpEXAMPLE_SG5Z-EXAMPLEPmPfEXAMPLEu3EqEXAMPLE=="
    },
    "viewer": {"ip": "198.51.100.11"},
```

```
    "request": {
        "method": "GET",
        "uri": "/media/index.mpd",
        "querystring": {
            "ID": {"value": "42"},
            "Exp": {"value": "1619740800"},
            "TTL": {"value": "1440"},
            "NoValue": {"value": ""},
            "querymv": {
                "value": "val1",
                "multiValue": [
                    {"value": "val1"},
                    {"value": "val2,val3"}
                ]
            }
        },
        "headers": {
            "host": {"value": "video.example.com"},
            "user-agent": {"value": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0)
 Gecko/20100101 Firefox/83.0"},
            "accept": {
                "value": "application/json",
                "multiValue": [
                    {"value": "application/json"},
                    {"value": "application/xml"},
                    {"value": "text/html"}
                ]
            },
            "accept-language": {"value": "en-GB,en;q=0.5"},
            "accept-encoding": {"value": "gzip, deflate, br"},
            "origin": {"value": "https://website.example.com"},
            "referer": {"value": "https://website.example.com/videos/12345678?
action=play"},
            "cloudfront-viewer-country": {"value": "GB"}
        },
        "cookies": {
            "Cookie1": {"value": "value1"},
            "Cookie2": {"value": "value2"},
            "cookie_consent": {"value": "true"},
            "cookiemv": {
                "value": "value3",
                "multiValue": [
                    {"value": "value3"},
                    {"value": "value4"}
```

```
                ]
            }
        }
    },
    "response": {
        "statusCode": 200,
        "statusDescription": "OK",
        "headers": {
            "date": {"value": "Mon, 04 Apr 2021 18:57:56 GMT"},
            "server": {"value": "gunicorn/19.9.0"},
            "access-control-allow-origin": {"value": "*"},
            "access-control-allow-credentials": {"value": "true"},
            "content-type": {"value": "application/json"},
            "content-length": {"value": "701"}
        },
        "cookies": {
            "ID": {
                "value": "id1234",
                "attributes": "Expires=Wed, 05 Apr 2021 07:28:00 GMT"
            },
            "Cookie1": {
                "value": "val1",
                "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr
 2021 07:28:00 GMT",
                "multiValue": [
                    {
                        "value": "val1",
                        "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed,
 05 Apr 2021 07:28:00 GMT"
                    },
                    {
                        "value": "val2",
                        "attributes": "Path=/cat; Domain=example.com; Expires=Wed, 10
 Jan 2021 07:28:00 GMT"
                    }
                ]
            }
        }
    }
}
```

CloudFront Functions の JavaScript ランタイムの機能

CloudFront Functions の JavaScript ランタイム環境は ECMAScript (ES) バージョン 5.1 に準拠しており、ES バージョン 6～12 の一部の機能をサポートしています。

最新の機能を利用するには、JavaScript ランタイム 2.0 を使用することをお勧めします。

JavaScript ランタイム 2.0 の機能には、1.0 と比較して以下の変更があります。

- バッファモジュールメソッドが利用可能です。
- 以下の非標準の文字列プロトタイプメソッドは使用できません。
  - `String.prototype.bytesFrom()`
  - `String.prototype.fromBytes()`
  - `String.prototype.fromUTF8()`
  - `String.prototype.toBytes()`
  - `String.prototype.toUTF8()`
- 暗号モジュールには次の変更があります。
  - `hash.digest()` - エンコーディングを指定しない場合、戻り型は `Buffer` に変更されます。
  - `hmac.digest()` - エンコーディングを指定しない場合、戻り型は `Buffer` に変更されます。
- その他の新しい機能の詳細については、「CloudFront Functions の JavaScript ランタイム 2.0 の機能」を参照してください。

トピック

- CloudFront Functions の JavaScript ランタイム 1.0 の機能
- CloudFront Functions の JavaScript ランタイム 2.0 の機能

CloudFront Functions の JavaScript ランタイム 1.0 の機能

CloudFront Functions の JavaScript ランタイム環境は ECMAScript (ES) バージョン 5.1 に準拠しており、ES バージョン 6～9 の一部の機能をサポートしています。また、ES 仕様に含まれない非標準メソッドも提供しています。

次のトピックでは、サポートされるすべての言語機能の一覧を示します。

トピック

- 主要機能

- プリミティブオブジェクト
- ビルトインオブジェクト
- エラーのタイプ
- Globals
- ビルトインモジュール
- 制限された機能

主要機能

ES の次の主要機能がサポートされています。

Types]

すべての ES 5.1 タイプでサポートされています。これには、ブール値、数値、文字列、オブジェクト、配列、関数、関数コンストラクタ、正規表現が含まれます。

演算子

すべての ES 5.1 演算子でサポートされています。

ES 7 指数演算子 (**) がサポートされています。

ステートメント

> **Note**
>
> `const` および `let` ステートメントはサポートされていません。

次の ES 5.1 ステートメントがサポートされています。

- `break`
- `catch`
- `continue`
- `do-while`
- `else`
- `finally`
- `for`

- for-in
- if
- return
- switch
- throw
- try
- var
- while
- ラベル付きステートメント

リテラル

ES 6 テンプレートリテラル (複数行の文字列、式の補間、および入れ子テンプレート) がサポートされています。

関数

すべての ES 5.1 機能がサポートされています。

ES 6 のアロー関数、ES 6 のレストパラメータ (残余因数) 構文がサポートされています。

Unicode

ソーステキストおよび文字列リテラルには、Unicode でエンコードされた文字を含めることができます。6 文字の Unicode エスケープシーケンス (コードポイント、例: \uXXXX) もサポートされています。

Strict モード

関数は Strict モードで動作するため、関数コードに `use strict` ステートメントを追加する必要はありません。これは変更できません。

プリミティブオブジェクト

以下の ES プリミティブオブジェクトがサポートされています。

オブジェクト

オブジェクトについて以下の ES 5.1 メソッドがサポートされています。

- create (プロパティリストなし)
- defineProperties
- defineProperty
- freeze
- getOwnPropertyDescriptor
- getOwnPropertyNames
- getPrototypeOf
- hasOwnProperty
- isExtensible
- isFrozen
- prototype.isPrototypeOf
- isSealed
- keys
- preventExtensions
- prototype.propertyIsEnumerable
- seal
- prototype.toString
- prototype.valueOf

オブジェクトについて以下の ES 6 メソッドがサポートされています。

- assign
- is
- prototype.setPrototypeOf

オブジェクトについて以下の ES 8 メソッドがサポートされています。

- entries
- values

文字列

文字列について以下の ES 5.1 メソッドがサポートされています。

- fromCharCode
- prototype.charAt
- prototype.concat
- prototype.indexOf
- prototype.lastIndexOf
- prototype.match
- prototype.replace
- prototype.search
- prototype.slice
- prototype.split
- prototype.substr
- prototype.substring
- prototype.toLowerCase
- prototype.trim
- prototype.toUpperCase

文字列について以下の ES 6 メソッドがサポートされています。

- fromCodePoint
- prototype.codePointAt
- prototype.endsWith
- prototype.includes
- prototype.repeat
- prototype.startsWith

文字列について以下の ES 8 メソッドがサポートされています。

- prototype.padStart
- prototype.padEnd

文字列について以下の ES 9 メソッドがサポートされています。

- prototype.trimStart

- prototype.trimEnd

文字列について以下の非標準メソッドがサポートされています。

- prototype.bytesFrom(*array | string, encoding*)

  オクテット列またはエンコードされた文字列からバイト文字列を作成します。文字列エンコーディングオプションは hex、base64、base64urlです。

- prototype.fromBytes(*start*[, *end*])

  バイト文字列から Unicode 文字列を作成します。各バイトは、対応する Unicode コードポイントで置き換えられます。

- prototype.fromUTF8(*start*[, *end*])

  UTF-8 でエンコードされたバイト文字列から Unicode 文字列を作成します。エンコーディングが正しくない場合は、null が返されます。

- prototype.toBytes(start[, end])

  Unicode 文字列からバイト文字列を作成します。すべての文字は [0,255] の範囲内にある必要があります。そうでない場合は、null が返されます。

- prototype.toUTF8(start[, *end*])

  Unicode 文字列から UTF-8 でエンコードされたバイト文字列を作成します。

数値

番号に関するすべての ES 5.1 メソッドがサポートされています。

番号について以下の ES 6 メソッドがサポートされています。

- isFinite
- isInteger
- isNaN
- isSafeInteger
- parseFloat
- parseInt
- prototype.toExponential
- prototype.toFixed

- prototype.toPrecision
- EPSILON
- MAX_SAFE_INTEGER
- MAX_VALUE
- MIN_SAFE_INTEGER
- MIN_VALUE
- NEGATIVE_INFINITY
- NaN
- POSITIVE_INFINITY

ビルトインオブジェクト

ES の以下のビルトインオブジェクトがサポートされています。

Math

ES 5.1 のすべての Math メソッドがサポートされています。

> **Note**
>
> CloudFront Functions runtime 環境では、Math.random() 実装に、関数が実行されたときのタイムスタンプがシードされた OpenBSD arc4random を使用します。

以下の ES 6 Math メソッドがサポートされています。

- acosh
- asinh
- atanh
- cbrt
- clz32
- cosh
- expm1
- fround

- hypot
- imul
- log10
- log1p
- log2
- sign
- sinh
- tanh
- trunc
- E
- LN10
- LN2
- LOG10E
- LOG2E
- PI
- SQRT1_2
- SQRT2

日付

すべての ES 5.1 の Date 機能がサポートされています。

> **Note**
>
> セキュリティ上の理由から、Date は、単一の関数実行の有効期間中、常に同じ値 (関数の開始時間) を返します。詳細については、「制限された機能」を参照してください。

関数

apply、bind、call メソッドがサポートされています。

関数コンストラクタはサポートされていません。

正規表現

すべての ES 5.1 の正規表現機能がサポートされています。正規表現言語は Perl 互換です。ES 9 の名前付きキャプチャグループがサポートされています。

JSON

parse、stringify を含むすべての ES 5.1 JSON 機能がサポートされています。

配列

配列について以下の ES 5.1 メソッドがサポートされています。

- isArray
- prototype.concat
- prototype.every
- prototype.filter
- prototype.forEach
- prototype.indexOf
- prototype.join
- prototype.lastIndexOf
- prototype.map
- prototype.pop
- prototype.push
- prototype.reduce
- prototype.reduceRight
- prototype.reverse
- prototype.shift
- prototype.slice
- prototype.some
- prototype.sort
- prototype.splice
- prototype.unshift

配列について以下の ES 6 メソッドがサポートされています。

- of
- prototype.copyWithin
- prototype.fill
- prototype.find
- prototype.findIndex

配列について以下の ES 7 メソッドがサポートされています。

- prototype.includes

型付き配列

以下の ES 6 型付き配列がサポートされています。

- Int8Array
- Uint8Array
- Uint8ClampedArray
- Int16Array
- Uint16Array
- Int32Array
- Uint32Array
- Float32Array
- Float64Array
- prototype.copyWithin
- prototype.fill
- prototype.join
- prototype.set
- prototype.slice
- prototype.subarray
- prototype.toString

ArrayBuffer

ArrayBuffer について以下のメソッドがサポートされています。

- `prototype.isView`
- `prototype.slice`

promise

promise について以下のメソッドがサポートされています。

- `reject`
- `resolve`
- `prototype.catch`
- `prototype.finally`
- `prototype.then`

Crypto

暗号モジュールは、標準のハッシュおよびハッシュベースのメッセージ認証コード (HMAC) ヘルパーを提供します。`require('crypto')` を使用してモジュールをロードできます。モジュールは、Node.js の相対物とまったく同じように動作する以下のメソッドを公開します。

- `createHash(algorithm)`
- `hash.update(data)`
- `hash.digest([encoding])`
- `createHmac(algorithm, secret key)`
- `hmac.update(data)`
- `hmac.digest([encoding])`

詳細については、「ビルトインモジュールセクション」の「Crypto (ハッシュと HMAC)」を参照してください。

コンソール

これはデバッグ用のヘルパーオブジェクトです。ログメッセージを記録するための `log()` メソッドのみサポートしています。

> **Note**
>
> CloudFront Functions は、`console.log('a', 'b')` などのカンマ構文をサポートしていません。代わりに、`console.log('a' + ' ' + 'b')` 形式を使用してください。

エラーのタイプ

以下のエラーオブジェクトがサポートされています。

- Error
- EvalError
- InternalError
- MemoryError
- RangeError
- ReferenceError
- SyntaxError
- TypeError
- URIError

Globals

globalThis オブジェクトはサポートされています。

以下の ES 5.1 グローバル関数がサポートされています。

- decodeURI
- decodeURIComponent
- encodeURI
- encodeURIComponent
- isFinite
- isNaN
- parseFloat
- parseInt

以下のグローバル定数がサポートされています。

- NaN
- Infinity
- undefined

ビルトインモジュール

以下のビルトインモジュールがサポートされています。

モジュール

- Crypto (ハッシュと HMAC)
- クエリ文字列

Crypto (ハッシュと HMAC)

暗号モジュール (crypto) は、標準のハッシュおよびハッシュベースのメッセージ認証コード (HMAC) ヘルパーを提供します。`require('crypto')` を使用してモジュールをロードできます。このモジュールは、Node.js の相対物とまったく同じように動作する以下のメソッドを提供します。

ハッシュメソッド

`crypto.createHash(`*`algorithm`*`)`

ハッシュオブジェクトを作成して返します。このハッシュオブジェクトは、指定されたアルゴリズム (md5、sha1、sha256 のいずれか) を使用してハッシュダイジェストの生成に使用できます。

`hash.update(`*`data`*`)`

指定された data を使用してハッシュコンテンツを更新します。

`hash.digest([`*`encoding`*`])`

hash.update() を使用して渡されたすべてのデータのダイジェストを計算します。エンコードは hex、base64、base64url のいずれかを使用します。

HMAC メソッド

`crypto.createHmac(`*`algorithm, secret key`*`)`

指定された algorithm と secret key を使用する HMAC オブジェクトを作成して返します。アルゴリズムは md5、sha1、sha256 のいずれかを使用します。

`hmac.update(`*`data`*`)`

指定された data を使用して HMAC コンテンツを更新します 。

`hmac.digest([encoding])`

`hmac.update()` を使用して渡されたすべてのデータのダイジェストを計算します。エンコードは `hex`、`base64`、`base64url` のいずれかを使用します。

クエリ文字列

> **Note**
>
> CloudFront Functions イベントオブジェクトは、URL クエリ文字列を自動的に解析します。つまり、ほとんどの場合、このモジュールを使用する必要はありません。

クエリ文字列モジュール (`querystring`) は、URL クエリ文字列を解析および書式設定するためのメソッドを提供します。`require('querystring')` を使用してモジュールをロードできます。このモジュールは、以下のメソッドを提供します。

`querystring.escape(string)`

URL は `string` をエンコードし、エスケープしたクエリ文字列を返します。このメソッドは `querystring.stringify()` で使用するため、直接使用しないでください。

`querystring.parse(string[, separator[, equal[, options]]])`

クエリ文字列 (`string`) を解析し、オブジェクトを返します。

`separator` パラメータは、クエリ文字列のキーと値のペアを区切る substring です。デフォルトでは、`&` です。

`equal` パラメータは、クエリ文字列のキーと値を区切る substring です。デフォルトでは、`=` です。

`options` パラメータは、以下のキーを持つオブジェクトです。

`decodeURIComponent` *function*

クエリ文字列のパーセントエンコーディングされた文字を decode する関数です。デフォルトでは、`querystring.unescape()` です。

`maxKeys` *number*

解析するキーの最大数。デフォルトでは、`1000` です。キーカウントの制限を解除するには、`0` の値を使用します。

デフォルトでは、クエリ文字列のパーセントエンコーディングされた文字は、UTF-8 エンコーディングを使用していると見なされます。無効な UTF-8 シーケンスは、U+FFFD 置換文字に置き換えられます。

たとえば、次のクエリ文字列の場合:

```
'name=value&abc=xyz&abc=123'
```

querystring.parse()の戻り値は次のとおりです。

```
{
name: 'value',
abc: ['xyz', '123']
}
```

querystring.decode() は querystring.parse() のエイリアスです。

querystring.stringify(*object*[, *separator*[, *equal*[, *options*]]])

object をシリアル化し、クエリ文字列を返します。

separator パラメータは、クエリ文字列のキーと値のペアを区切る substring です。デフォルトでは、& です。

equal パラメータは、クエリ文字列のキーと値を区切る substring です。デフォルトでは、= です。

options パラメータは、以下のキーを持つオブジェクトです。

encodeURIComponent *function*

URL-unsafe 文字をクエリ文字列のパーセントエンコーディングに変換するために使用される関数です。デフォルトでは、querystring.escape() です。

デフォルトでは、クエリ文字列でパーセントエンコーディングが必要な文字は UTF-8 としてエンコードされます。別のエンコーディングを使用するには、encodeURIComponent オプションを指定します。

以下のコードでの例:

```
querystring.stringify({ name: 'value', abc: ['xyz', '123'], anotherName: '' });
```

戻り値:

```
'name=value&abc=xyz&abc=123&anotherName='
```

querystring.encode() は querystring.stringify() のエイリアスです。

querystring.unescape(*string*)

指定された string 内の URL パーセントエンコーディングされた文字をデコードし、エスケープしていないクエリ文字列を返します。このメソッドは querystring.parse()で使用するため、直接使用しないでください。

制限された機能

次の JavaScript 言語機能は、セキュリティ上の問題により、サポートされていないか、制限されています。

動的コード評価

動的コード評価はサポートされていません。eval()、Function 両方のコンストラクタが試行された場合、エラーをスローします。たとえば、const sum = new Function('a', 'b', 'return a + b') はエラーをスローします。

タイマー

setTimeout()、setImmediate()、clearTimeout() 関数はサポートされていません。関数実行中に defer または yield する規定はありません。関数は同期的に実行しないと完了できません。

日付とタイムスタンプ

セキュリティ上の理由から、高解像度タイマーにはアクセスできません。現在の時刻を照会するすべての Date メソッドは、単一の関数実行の存続期間中は常に同じ値を返します。返されるタイムスタンプは、関数の実行を開始した時刻です。したがって、関数内で経過時間を測定することはできません。

ファイルシステムへのアクセス

ファイルシステムにはアクセスできません。たとえば、Node.js にあるようなファイルシステムアクセス用の fs モジュールはありません。

ネットワークアクセス

ネットワークコールはサポートされていません。たとえば、XHR、HTTP (S)、ソケットはサポートされていません。

CloudFront Functions の JavaScript ランタイム 2.0 の機能

CloudFront Functions の JavaScript ランタイム環境は ECMAScript (ES) バージョン 5.1 に準拠しており、ES バージョン 6～12 の一部の機能をサポートしています。また、ES 仕様に含まれない非標準メソッドも提供しています。次のトピックでは、このランタイムでサポートされるすべての機能を一覧表示します。

トピック

主要機能

ES の次の主要機能がサポートされています。

Types]

すべての ES 5.1 タイプでサポートされています。これには、ブール値、数値、文字列、オブジェクト、配列、関数、正規表現が含まれます。

演算子

すべての ES 5.1 演算子でサポートされています。

ES 7 指数演算子 (**) がサポートされています。

ステートメント

次の ES 5.1 ステートメントがサポートされています。

- `break`
- `catch`
- `continue`

- do-while
- else
- finally
- for
- for-in
- if
- label
- return
- switch
- throw
- try
- var
- while

次の ES 6 ステートメントがサポートされています。

- async
- await
- const
- let

> **Note**
>
> async、await、const、let は JavaScript ランタイム 2.0 で新しく追加されました。

リテラル

ES 6 テンプレートリテラル (複数行の文字列、式の補間、および入れ子テンプレート) がサポートされています。

関数

すべての ES 5.1 機能がサポートされています。

ES 6 のアロー関数、ES 6 のレストパラメータ (残余因数) 構文がサポートされています。

Unicode

ソーステキストおよび文字列リテラルには、Unicode でエンコードされた文字を含めることができます。6 文字の Unicode エスケープシーケンス (コードポイント、例: \uXXXX) もサポートされています。

Strict モード

関数は Strict モードで動作するため、関数コードに `use strict` ステートメントを追加する必要はありません。これは変更できません。

プリミティブオブジェクト

以下の ES プリミティブオブジェクトがサポートされています。

オブジェクト

オブジェクトについて以下の ES 5.1 メソッドがサポートされています。

- `Object.create()` (プロパティリストなし)
- `Object.defineProperties()`
- `Object.defineProperty()`
- `Object.freeze()`
- `Object.getOwnPropertyDescriptor()`
- `Object.getOwnPropertyDescriptors()`
- `Object.getOwnPropertyNames()`
- `Object.getPrototypeOf()`
- `Object.isExtensible()`
- `Object.isFrozen()`
- `Object.isSealed()`
- `Object.keys()`
- `Object.preventExtensions()`
- `Object.seal()`

オブジェクトについて以下の ES 6 メソッドがサポートされています。

- `Object.assign()`

オブジェクトについて以下の ES 8 メソッドがサポートされています。

- Object.entries()
- Object.values()

オブジェクトについて以下の ES 5.1 プロトタイプメソッドがサポートされています。

- Object.prototype.hasOwnProperty()
- Object.prototype.isPrototypeOf()
- Object.prototype.propertyIsEnumerable()
- Object.prototype.toString()
- Object.prototype.valueOf()

オブジェクトについて以下の ES 6 プロトタイプメソッドがサポートされています。

- Object.prototype.is()
- Object.prototype.setPrototypeOf()

文字列

文字列について以下の ES 5.1 メソッドがサポートされています。

- String.fromCharCode()

文字列について以下の ES 6 メソッドがサポートされています。

- String.fromCodePoint()

文字列について以下の ES 5.1 プロトタイプメソッドがサポートされています。

- String.prototype.charAt()
- String.prototype.concat()
- String.prototype.indexOf()
- String.prototype.lastIndexOf()
- String.prototype.match()
- String.prototype.replace()
- String.prototype.search()
- String.prototype.slice()
- String.prototype.split()
- String.prototype.substr()
- String.prototype.substring()
- String.prototype.toLowerCase()

- String.prototype.trim()
- String.prototype.toUpperCase()

文字列について以下の ES 6 プロトタイプメソッドがサポートされています。

- String.prototype.codePointAt()
- String.prototype.endsWith()
- String.prototype.includes()
- String.prototype.repeat()
- String.prototype.startsWith()

文字列について以下の ES 8 プロトタイプメソッドがサポートされています。

- String.prototype.padStart()
- String.prototype.padEnd()

文字列について以下の ES 9 プロトタイプメソッドがサポートされています。

- String.prototype.trimStart()
- String.prototype.trimEnd()

文字列について以下の ES 12 プロトタイプメソッドがサポートされています。

- String.prototype.replaceAll()

> **Note**
>
> String.prototype.replaceAll() は JavaScript ランタイム 2.0 で新しく追加されました。

数

すべての ES 5 番号がサポートされています。

番号について以下の ES 6 プロパティがサポートされています。

- Number.EPSILON
- Number.MAX_SAFE_INTEGER
- Number.MIN_SAFE_INTEGER
- Number.MAX_VALUE
- Number.MIN_VALUE

- `Number.NaN`
- `Number.NEGATIVE_INFINITY`
- `Number.POSITIVE_INFINITY`

番号について以下の ES 6 メソッドがサポートされています。

- `Number.isFinite()`
- `Number.isInteger()`
- `Number.isNaN()`
- `Number.isSafeInteger()`
- `Number.parseInt()`
- `Number.parseFloat()`

番号について以下の ES 5.1 プロトタイプメソッドがサポートされています。

- `Number.prototype.toExponential()`
- `Number.prototype.toFixed()`
- `Number.prototype.toPrecision()`

ES 12 数字区切り文字がサポートされています。

> **Note**
>
> ES 12 数字区切り文字は JavaScript ランタイム 2.0 で新しく追加されました。

ビルトインオブジェクト

ES の以下のビルトインオブジェクトがサポートされています。

Math

ES 5.1 のすべての Math メソッドがサポートされています。

> **Note**
>
> CloudFront Functions runtime 環境では、`Math.random()` 実装に、関数が実行されたときのタイムスタンプがシードされた OpenBSD `arc4random` を使用します。

以下の ES 6 数学的プロパティがサポートされています。

- Math.E
- Math.LN10
- Math.LN2
- Math.LOG10E
- Math.LOG2E
- Math.PI
- Math.SQRT1_2
- Math.SQRT2

以下の ES 6 Math メソッドがサポートされています。

- Math.abs()
- Math.acos()
- Math.acosh()
- Math.asin()
- Math.asinh()
- Math.atan()
- Math.atan2()
- Math.atanh()
- Math.cbrt()
- Math.ceil()
- Math.clz32()
- Math.cos()
- Math.cosh()
- Math.exp()
- Math.expm1()
- Math.floor()
- Math.fround()
- Math.hypot()
- Math.imul()

- Math.log()
- Math.log1p()
- Math.log2()
- Math.log10()
- Math.max()
- Math.min()
- Math.pow()
- Math.random()
- Math.round()
- Math.sign()
- Math.sinh()
- Math.sin()
- Math.sqrt()
- Math.tan()
- Math.tanh()
- Math.trunc()

日付

すべての ES 5.1 の Date 機能がサポートされています。

> **Note**
>
> セキュリティ上の理由から、Date は、単一の関数実行の有効期間中、常に同じ値 (関数の開始時間) を返します。詳細については、「制限された機能」を参照してください。

関数

以下の ES 5.1 プロトタイプメソッドがサポートされています。

- Function.prototype.apply()
- Function.prototype.bind()
- Function.prototype.call()

関数コンストラクタはサポートされていません。

正規表現

すべての ES 5.1 の正規表現機能がサポートされています。正規表現言語は Perl 互換です。

以下の ES 5.1 プロトタイプアクセサプロパティがサポートされています。

- `RegExp.prototype.global`
- `RegExp.prototype.ignoreCase`
- `RegExp.protoype.multiline`
- `RegExp.protoype.source`
- `RegExp.prototype.sticky`
- `RegExp.prototype.flags`

> **Note**
>
> `RegExp.prototype.sticky` および `RegExp.prototype.flags` は JavaScript ランタイム 2.0 で新しく追加されました。

以下の ES 5.1 プロトタイプメソッドがサポートされています。

- `RegExp.prototype.exec()`
- `RegExp.prototype.test()`
- `RegExp.prototype.toString()`
- `RegExp.prototype[@@replace]()`
- `RegExp.prototype[@@split]()`

> **Note**
>
> `RegExp.prototype[@@split]()` は JavaScript ランタイム 2.0 で新しく追加されました。

以下の ES 5.1 インスタンスプロパティがサポートされています。

- `lastIndex`

ES 9 の名前付きキャプチャグループがサポートされています。

JSON

以下の ES 5.1 メソッドがサポートされています。

- JSON.parse()
- JSON.stringify()

配列

配列について以下の ES 5.1 メソッドがサポートされています。

- Array.isArray()

配列について以下の ES 6 メソッドがサポートされています。

- Array.of()

以下の ES 5.1 プロトタイプメソッドがサポートされています。

- Array.prototype.concat()
- Array.prototype.every()
- Array.prototype.filter()
- Array.prototype.forEach()
- Array.prototype.indexOf()
- Array.prototype.join()
- Array.prototype.lastIndexOf()
- Array.prototype.map()
- Array.prototype.pop()
- Array.prototype.push()
- Array.prototype.reduce()
- Array.prototype.reduceRight()
- Array.prototype.reverse()
- Array.prototype.shift()
- Array.prototype.slice()
- Array.prototype.some()
- Array.prototype.sort()
- Array.prototype.splice()
- Array.prototype.unshift()

以下の ES 6 プロトタイプメソッドがサポートされています。

- Array.prototype.copyWithin()

- Array.prototype.fill()
- Array.prototype.find()
- Array.prototype.findIndex()

以下の ES 7 プロトタイプメソッドがサポートされています。

- Array.prototype.includes()

型付き配列

以下の ES 6 型付き配列コンストラクターがサポートされています。

- Float32Array
- Float64Array
- Int8Array
- Int16Array
- Int32Array
- Uint8Array
- Uint8ClampedArray
- Uint16Array
- Uint32Array

以下の ES 6 メソッドがサポートされています。

- TypedArray.from()
- TypedArray.of()

> **Note**
>
> TypedArray.from() および TypedArray.of() は JavaScript ランタイム 2.0 で新しく追加されました。

以下の ES 6 プロトタイプメソッドがサポートされています。

- TypedArray.prototype.copyWithin()
- TypedArray.prototype.every()
- TypedArray.prototype.fill()
- TypedArray.prototype.filter()

- TypedArray.prototype.find()
- TypedArray.prototype.findIndex()
- TypedArray.prototype.forEach()
- TypedArray.prototype.includes()
- TypedArray.prototype.indexOf()
- TypedArray.prototype.join()
- TypedArray.prototype.lastIndexOf()
- TypedArray.prototype.map()
- TypedArray.prototype.reduce()
- TypedArray.prototype.reduceRight()
- TypedArray.prototype.reverse()
- TypedArray.prototype.some()
- TypedArray.prototype.set()
- TypedArray.prototype.slice()
- TypedArray.prototype.sort()
- TypedArray.prototype.subarray()
- TypedArray.prototype.toString()

> **Note**
>
> TypedArray.prototype.every()、TypedArray.prototype.fill()、TypedArray.pro
> および TypedArray.prototype.some() は JavaScript ランタイム 2.0 で新しく追加
> されました。

ArrayBuffer

ArrayBuffer について以下の ES 6 メソッドがサポートされています。

- isView()

ArrayBuffer について以下の ES 6 プロトタイプメソッドがサポートされています。

- ArrayBuffer.prototype.slice()

promise

Promise について以下の ES 6 メソッドがサポートされています。

- `Promise.all()`
- `Promise.allSettled()`
- `Promise.any()`
- `Promise.reject()`
- `Promise.resolve()`
- `Promise.race()`

> **Note**
>
> `Promise.all()`、`Promise.allSettled()`、`Promise.any()`、`Promise.race()` は JavaScript ランタイム 2.0 で新しく追加されました。

Promise について以下の ES 6 プロトタイプメソッドがサポートされています。

- `Promise.prototype.catch()`
- `Promise.prototype.finally()`
- `Promise.prototype.then()`

DataView

以下の ES 6 プロトタイプメソッドがサポートされています。

- `DataView.prototype.getFloat32()`
- `DataView.prototype.getFloat64()`
- `DataView.prototype.getInt16()`
- `DataView.prototype.getInt32()`
- `DataView.prototype.getInt8()`
- `DataView.prototype.getUint16()`
- `DataView.prototype.getUint32()`
- `DataView.prototype.getUint8()`
- `DataView.prototype.setFloat32()`
- `DataView.prototype.setFloat64()`
- `DataView.prototype.setInt16()`
- `DataView.prototype.setInt32()`
- `DataView.prototype.setInt8()`

- `DataView.prototype.setUint16()`
- `DataView.prototype.setUint32()`
- `DataView.prototype.setUint8()`

> **Note**
>
> Dataview ES 6 のプロトタイプメソッドはすべて JavaScript ランタイム 2.0 で新しく追加されました。

記号

以下の ES 6 メソッドがサポートされています。

- `Symbol.for()`
- `Symbol.keyfor()`

> **Note**
>
> Symbol ES 6 メソッドはすべて JavaScript ランタイム 2.0 で新しく追加されました。

テキストデコーダー

以下のプロトタイプメソッドがサポートされています。

- `TextDecoder.prototype.decode()`

以下のプロトタイプアクセサプロパティがサポートされています。

- `TextDecoder.prototype.encoding`
- `TextDecoder.prototype.fatal`
- `TextDecoder.prototype.ignoreBOM`

テキストエンコーダー

以下のプロトタイプメソッドがサポートされています。

- `TextEncoder.prototype.encode()`
- `TextEncoder.prototype.encodeInto()`

エラーのタイプ

以下のエラーオブジェクトがサポートされています。

- Error
- EvalError
- InternalError
- RangeError
- ReferenceError
- SyntaxError
- TypeError
- URIError

Globals

globalThis オブジェクトはサポートされています。

以下の ES 5.1 グローバル関数がサポートされています。

- decodeURI()
- decodeURIComponent()
- encodeURI()
- encodeURIComponent()
- isFinite()
- isNaN()
- parseFloat()
- parseInt()

以下の ES 6 グローバル関数がサポートされています。

- atob()
- btoa()

> **Note**
>
> atob() および btoa() は JavaScript ランタイム 2.0 で新しく追加されました。

以下のグローバル定数がサポートされています。

- `NaN`
- `Infinity`
- `undefined`
- `arguments`

ビルトインモジュール

以下のビルトインモジュールがサポートされています。

モジュール

- バッファ
- クエリ文字列
- Crypto

バッファ

このモジュールは、以下のメソッドを提供します。

- `Buffer.alloc(size[, fill[, encoding]])`

  `Buffer` を割り当てます。

  - `size`: バッファサイズ。整数を入力します。
  - `fill`: オプション。文字列、`Buffer`、Uint8Array または整数を入力します。デフォルトは `0` です。
  - `encoding`: オプション。`fill` が文字列である場合は、`utf8`、`hex`、`base64`、`base64url` のいずれかを入力します。デフォルトは `utf8` です。
- `Buffer.allocUnsafe(size)`

  初期化されていない `Buffer` を割り当てます。

  - `size`: 整数を入力します。
- `Buffer.byteLength(value[, encoding])`

  値の長さをバイト単位で返します。

  - `value`: 文字列、`Buffer`、TypedArray、Dataview、または Arraybuffer。

  - encoding: オプション。value が文字列である場合は、utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。
- Buffer.compare(buffer1, buffer2)

  2 つの Buffer を比較すると、配列をソートしやすくなります。両者が同じ場合は 0、buffer1 が先に来る場合は -1、buffer2 が先に来る場合は 1 を返します。

  - buffer1: Buffer を入力します。
  - buffer2: 別の Buffer 値を入力します。
- Buffer.concat(list[, totalLength])

  複数の Buffer を連結します。ない場合は 0 を返します。totalLength までの値を返します。

  - list: Buffer のリストを入力します。これは totalLength に切り捨てられることに注意してください。
  - totalLength: オプション。符号なし整数を入力します。空欄の場合はリスト内の Buffer インスタンス総数を使用します。
- Buffer.from(array)

  配列から Buffer を作成します。

  - array: 0 から 255 までのバイト配列を入力します。
- Buffer.from(arrayBuffer, byteOffset[, length]))

  オフセット byteOffset から始めて長さが length のビューを arrayBuffer から作成します。

  - arrayBuffer: Buffer 配列を入力します。
  - byteOffset: 整数を入力します。
  - length: オプション。整数を入力します。
- Buffer.from(buffer)

  Buffer のコピーを作成します。

  - buffer: Buffer を入力します。
- Buffer.from(object[, offsetOrEncoding[, length]])

  オブジェクトから Buffer を作成します。valueOf() がオブジェクトと等しくない場合は Buffer.from(object.valueOf(), offsetOrEncoding, length) を返します。

  - object: オブジェクトを入力します。
  - offsetOrEncoding: オプション。整数またはエンコーディング文字列を入力します。

  - length: オプション。整数を入力します。
- Buffer.from(string[, encoding])

  文字列から Buffer を作成します。

  - string: 文字列を入力します。
  - encoding: オプション。utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。
- Buffer.isBuffer(object)

  object がバッファかどうかをチェックします。true または false を返します。

  - object: オブジェクトを入力します。
- Buffer.isEncoding(encoding)

  encoding がサポートされているかをチェックします。true または false を返します。

  - encoding: オプション。utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。

このモジュールは、以下のバッファプロトタイプメソッドを提供します。

- Buffer.prototype.compare(target[, targetStart[, targetEnd[, sourceStart[, sourceEnd]]]])

  ターゲットと Buffer を比較します。両者が同じ場合は 0、buffer が先に来る場合は 1、target が先に来る場合は -1 を返します。

  - target: Buffer を入力します。
  - targetStart: オプション。整数を入力します。デフォルトは 0 です。
  - targetEnd: オプション。整数を入力します。デフォルトは target の長さです。
  - sourceStart: オプション。整数を入力します。デフォルトは 0 です。
  - sourceEnd: オプション。整数を入力します。デフォルトは Buffer の長さです。
- Buffer.prototype.copy(target[, targetStart[, sourceStart[, sourceEnd]]])

  バッファを target にコピーします。

  - target: Buffer または Uint8Array を入力します。
  - targetStart: オプション。整数を入力します。デフォルトは 0 です。
  - sourceStart: オプション。整数を入力します。デフォルトは 0 です。

  - sourceEnd: オプション。整数を入力します。デフォルトは Buffer の長さです。
- Buffer.prototype.equals(otherBuffer)

  Buffer と otherBuffer を比較します。true または false を返します。

  - otherBuffer: 文字列を入力します。
- Buffer.prototype.fill(value[, offset[, end][, encoding])

  value に Buffer を入力します。

  - value: 文字列、Buffer、または整数を入力します。
  - offset: オプション。整数を入力します。
  - end: オプション。整数を入力します。
  - encoding: オプション。utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。
- Buffer.prototype.includes(value[, byteOffset][, encoding])

  value で Buffer を検索します。true または false を返します。

  - value: 文字列、Buffer、Uint8Array、または整数を入力します。
  - byteOffset: オプション。整数を入力します。
  - encoding: オプション。utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。
- Buffer.prototype.indexOf(value[, byteOffset][, encoding])

  Buffer で最初の value を検索します。見つかった場合は index を返し、見つからなかった場合は -1 を返します。

  - value: 文字列、Buffer、Unit8Array、または 0 から 255 までの整数を入力します。
  - byteOffset: オプション。整数を入力します。
  - encoding: オプション。value が文字列の場合、utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。
- Buffer.prototype.lastIndexOf(value[, byteOffset][, encoding])

  Buffer で最後の value を検索します。見つかった場合は index を返し、見つからなかった場合は -1 を返します。

  - value: 文字列、Buffer、Unit8Array、または 0 から 255 までの整数を入力します。
  - byteOffset: オプション。整数を入力します。

  - encoding: オプション。value が文字列の場合、utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。
- Buffer.prototype.readInt8(offset)

  Buffer から offset で Int8 を読み込みます。

  - offset: 整数を入力します。
- Buffer.prototype.readIntBE(offset, byteLength)

  Buffer から offset で Int をビッグエンディアンとして読み取ります。

  - offset: 整数を入力します。
  - byteLength: オプション。1 から 6 までの整数を入力します。
- Buffer.prototype.readInt16BE(offset)

  Buffer から offset で Int16 をビッグエンディアンとして読み取ります。

  - offset: 整数を入力します。
- Buffer.prototype.readInt32BE(offset)

  Buffer から offset で Int32 をビッグエンディアンとして読み取ります。

  - offset: 整数を入力します。
- Buffer.prototype.readIntLE(offset, byteLength)

  Buffer から offset で Int をリトルエンディアンとして読み取ります。

  - offset: 整数を入力します。
  - byteLength: 1 から 6 までの整数を入力します。
- Buffer.prototype.readInt16LE(offset)

  Buffer から offset で Int16 をリトルエンディアンとして読み取ります。

  - offset: 整数を入力します。
- Buffer.prototype.readInt32LE(offset)

  Buffer から offset で Int32 をリトルエンディアンとして読み取ります。

  - offset: 整数を入力します。
- Buffer.prototype.readUInt8(offset)

  Buffer から offset で UInt8 を読み込みます。

  - offset: 整数を入力します。
- Buffer.prototype.readUIntBE(offset, byteLength)

  Buffer から offset で UInt をビッグエンディアンとして読み取ります。

  - offset: 整数を入力します。
  - byteLength: 1 から 6 までの整数を入力します。
- Buffer.prototype.readUInt16BE(offset)

  Buffer から offset で UInt16 をビッグエンディアンとして読み取ります。
- • offset: 整数を入力します。
- Buffer.prototype.readUInt32BE(offset)

  Buffer から offset で UInt32 をビッグエンディアンとして読み取ります。

  - offset: 整数を入力します。
- Buffer.prototype.readUIntLE(offset, byteLength)

  Buffer から offset で UInt をリトルエンディアンとして読み取ります。

  - offset: 整数を入力します。
  - byteLength: 1 から 6 までの整数を入力します。
- Buffer.prototype.readUInt16LE(offset)

  Buffer から offset で UInt16 をリトルエンディアンとして読み取ります。

  - offset: 整数を入力します。
- Buffer.prototype.readUInt32LE(offset)

  Buffer から offset で UInt32 をリトルエンディアンとして読み取ります。

  - offset: 整数を入力します。
- Buffer.prototype.readDoubleBE([offset])

  Buffer から offset で 64 ビットダブルをビッグエンディアンとして読み込みます。

  - offset: オプション。整数を入力します。
- Buffer.prototype.readDoubleLE([offset])

  Buffer から offset で 64 リトルダブルをビッグエンディアンとして読み込みます。

  - offset: オプション。整数を入力します。

- Buffer.prototype.readFloatBE([offset])

  Buffer から offset で 32 ビットフロートをビッグエンディアンとして読み込みます。

  - offset: オプション。整数を入力します。
- Buffer.prototype.readFloatLE([offset])

  Buffer から offset で 32 ビットフロートをリトルエンディアンとして読み込みます。

  - offset: オプション。整数を入力します。
- Buffer.prototype.subarray([start[, end]])

  オフセットし、新しい start および end で切り取った Buffer のコピーを返します。

  - start: オプション。整数を入力します。デフォルトは 0 です。
  - end: オプション。整数を入力します。デフォルトはバッファの長さです。
- Buffer.prototype.swap16()

  Buffer 配列のバイト順を入れ替え、16 ビットの数値の配列として扱います。Buffer の長さは 2 で割り切れる必要があります。そうしないと、エラーになります。
- Buffer.prototype.swap32()

  Buffer 配列のバイト順を入れ替え、32 ビットの数値の配列として扱います。Buffer の長さは 4 で割り切れる必要があります。そうしないと、エラーになります。
- Buffer.prototype.swap64()

  Buffer 配列のバイト順を入れ替え、64 ビットの数値の配列として扱います。Buffer の長さは 8 で割り切れる必要があります。そうしないと、エラーになります。
- Buffer.prototype.toJSON()

  JSON として Buffer を返します。
- Buffer.prototype.toString([encoding[, start[, end]]])

  start から end まで Buffer をエンコードされた文字列に変換します。

  - encoding: オプション。utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。
  - start: オプション。整数を入力します。デフォルトは 0 です。
  - end: オプション。整数を入力します。デフォルトはバッファの長さです。
- Buffer.prototype.write(string[, offset[, length]][, encoding])

スペースがある場合はエンコードされた string を Buffer に書き込み、十分なスペースがない場合は切り捨てられた string になります。

- string: 文字列を入力します。
- offset: オプション。整数を入力します。デフォルトは 0 です。
- length: オプション。整数を入力します。デフォルトは文字列の長さです。
- encoding: オプション。オプションで、utf8、hex、base64、または base64url のいずれかを入力します。デフォルトは utf8 です。

- Buffer.prototype.writeInt8(value, offset, byteLength)

  offset で byteLength の Int8 value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: 整数を入力します
  - byteLength: 1 から 6 までの整数を入力します。
- Buffer.prototype.writeIntBE(value, offset, byteLength)

  ビッグエンディアンを使用して offset の value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: 整数を入力します
  - byteLength: 1 から 6 までの整数を入力します。
- Buffer.prototype.writeInt16BE(value, offset, byteLength)

  ビッグエンディアンを使用して offset の value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: 整数を入力します
  - byteLength: 1 から 6 までの整数を入力します。
- Buffer.prototype.writeInt32BE(value, offset, byteLength)

  ビッグエンディアンを使用して offset の value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: 整数を入力します
  - byteLength: 1 から 6 までの整数を入力します。
- Buffer.prototype.writeIntLE(offset, byteLength)

リトルエンディアンを使用して offset の value を Buffer に書き込みます。

- offset: 整数を入力します。
- byteLength: 1 から 6 までの整数を入力します。

- Buffer.prototype.writeInt16LE(offset, byteLength)

  リトルエンディアンを使用して offset の value を Buffer に書き込みます。

  - offset: 整数を入力します。
  - byteLength: 1 から 6 までの整数を入力します。

- Buffer.prototype.writeInt32LE(offset, byteLength)

  リトルエンディアンを使用して offset の value を Buffer に書き込みます。

  - offset: 整数を入力します。
  - byteLength: 1 から 6 までの整数を入力します。

- Buffer.prototype.writeUInt8(value, offset, byteLength)

  offset で byteLength の UInt8 value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: 整数を入力します
  - byteLength: 1 から 6 までの整数を入力します。

- Buffer.prototype.writeUIntBE(value, offset, byteLength)

  ビッグエンディアンを使用して offset の value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: 整数を入力します
  - byteLength: 1 から 6 までの整数を入力します。

- Buffer.prototype.writeUInt16BE(value, offset, byteLength)

  ビッグエンディアンを使用して offset の value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: 整数を入力します
  - byteLength: 1 から 6 までの整数を入力します。

- Buffer.prototype.writeUInt32BE(value, offset, byteLength)

ビッグエンディアンを使用して offset の value を Buffer に書き込みます。

- value: 整数を入力します。
- offset: 整数を入力します
- byteLength: 1 から 6 までの整数を入力します。

- Buffer.prototype.writeUIntLE(value, offset, byteLength)

  リトルエンディアンを使用して offset の value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: 整数を入力します
  - byteLength: 1 から 6 までの整数を入力します。
- Buffer.prototype.writeUInt16LE(value, offset, byteLength)

  リトルエンディアンを使用して offset の value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: 整数を入力します
  - byteLength: 1 から 6 までの整数を入力します。
- Buffer.prototype.writeUInt32LE(value, offset, byteLength)

  リトルエンディアンを使用して offset の value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: 整数を入力します
  - byteLength: 1 から 6 までの整数を入力します。
- Buffer.prototype.writeDoubleBE(value, [offset])

  ビッグエンディアンを使用して offset の value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: オプション。整数を入力します。デフォルトは 0 です。
- Buffer.prototype.writeDoubleLE(value, [offset])

  リトルエンディアンを使用して offset の value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: オプション。整数を入力します。デフォルトは 0 です。
- Buffer.prototype.writeFloatBE(value, [offset])

ビッグエンディアンを使用して offset の value を Buffer に書き込みます。

- value: 整数を入力します。
- offset: オプション。整数を入力します。デフォルトは 0 です。

- Buffer.prototype.writeFloatLE(value, [offset])

  リトルエンディアンを使用して offset の value を Buffer に書き込みます。

  - value: 整数を入力します。
  - offset: オプション。整数を入力します。デフォルトは 0 です。

以下のインスタンスメソッドがサポートされています。

- buffer[index]

  Buffer index でオクテット (バイト) を取得および設定します。

  - 0 から 255 までの数値を取得します。または、0 から 255 までの数値を設定します。

以下のインスタンスプロパティがサポートされています。

- buffer

  バッファの ArrayBuffer オブジェクトを取得します。

- byteOffset

  バッファの Arraybuffer オブジェクトの byteOffset を取得します。

- length

  バッファのバイト数を取得します。

> **Note**
> バッファモジュールメソッドはすべて JavaScript ランタイム 2.0 で新しく追加されました。

クエリ文字列

> **Note**
>
> CloudFront Functions イベントオブジェクトは、URL クエリ文字列を自動的に解析します。つまり、ほとんどの場合、このモジュールを使用する必要はありません。

クエリ文字列モジュール (querystring) は、URL クエリ文字列を解析および書式設定するためのメソッドを提供します。`require('querystring')` を使用してモジュールをロードできます。このモジュールは、以下のメソッドを提供します。

`querystring.escape(`*`string`*`)`

URL は `string` をエンコードし、エスケープしたクエリ文字列を返します。このメソッドは `querystring.stringify()` で使用するため、直接使用しないでください。

`querystring.parse(`*`string`*`[, `*`separator`*`[, `*`equal`*`[, `*`options`*`]]])`

クエリ文字列 (`string`) を解析し、オブジェクトを返します。

`separator` パラメータは、クエリ文字列のキーと値のペアを区切る substring です。デフォルトでは、`&` です。

`equal` パラメータは、クエリ文字列のキーと値を区切る substring です。デフォルトでは、`=` です。

`options` パラメータは、以下のキーを持つオブジェクトです。

`decodeURIComponent` *`function`*

クエリ文字列のパーセントエンコーディングされた文字を decode する関数です。デフォルトでは、`querystring.unescape()` です。

`maxKeys` *`number`*

解析するキーの最大数。デフォルトでは、`1000` です。キーカウントの制限を解除するには、`0` の値を使用します。

デフォルトでは、クエリ文字列のパーセントエンコーディングされた文字は、UTF-8 エンコーディングを使用していると見なされます。無効な UTF-8 シーケンスは、`U+FFFD` 置換文字に置き換えられます。

たとえば、次のクエリ文字列の場合:

```
'name=value&abc=xyz&abc=123'
```

querystring.parse()の戻り値は次のとおりです。

```
{
name: 'value',
abc: ['xyz', '123']
}
```

querystring.decode() は querystring.parse() のエイリアスです。

querystring.stringify(*object*[, *separator*[, *equal*[, *options*]]])

object をシリアル化し、クエリ文字列を返します。

separator パラメータは、クエリ文字列のキーと値のペアを区切る substring です。デフォルトでは、& です。

equal パラメータは、クエリ文字列のキーと値を区切る substring です。デフォルトでは、= です。

options パラメータは、以下のキーを持つオブジェクトです。

encodeURIComponent *function*

URL-unsafe 文字をクエリ文字列のパーセントエンコーディングに変換するために使用される関数です。デフォルトでは、querystring.escape() です。

デフォルトでは、クエリ文字列でパーセントエンコーディングが必要な文字は UTF-8 としてエンコードされます。別のエンコーディングを使用するには、encodeURIComponent オプションを指定します。

以下のコードでの例:

```
querystring.stringify({ name: 'value', abc: ['xyz', '123'], anotherName: '' });
```

戻り値:

```
'name=value&abc=xyz&abc=123&anotherName='
```

querystring.encode() は querystring.stringify() のエイリアスです。

querystring.unescape(*string*)

指定された string 内の URL パーセントエンコーディングされた文字をデコードし、エスケープしていないクエリ文字列を返します。このメソッドは querystring.parse()で使用するため、直接使用しないでください。

Crypto

暗号モジュール (crypto) は、標準のハッシュおよびハッシュベースのメッセージ認証コード (HMAC) ヘルパーを提供します。require('crypto') を使用してモジュールをロードできます。

ハッシュメソッド

crypto.createHash(*algorithm*)

ハッシュオブジェクトを作成して返します。このハッシュオブジェクトは、指定されたアルゴリズム (md5、sha1、sha256 のいずれか) を使用してハッシュダイジェストの生成に使用できます。

hash.update(*data*)

指定された data を使用してハッシュコンテンツを更新します。

hash.digest([*encoding*])

hash.update() を使用して渡されたすべてのデータのダイジェストを計算します。エンコードは hex、base64、base64url のいずれかを使用します。

HMAC メソッド

crypto.createHmac(*algorithm, secret key*)

指定された algorithm と secret key を使用する HMAC オブジェクトを作成して返します。アルゴリズムは md5、sha1、sha256 のいずれかを使用します。

hmac.update(*data*)

指定された data を使用して HMAC コンテンツを更新します 。

hmac.digest([*encoding*])

hmac.update() を使用して渡されたすべてのデータのダイジェストを計算します。エンコードは hex、base64、base64url のいずれかを使用します。

制限された機能

次の JavaScript 言語機能は、セキュリティ上の問題により、サポートされていないか、制限されています。

動的コード評価

動的コード評価はサポートされていません。`eval()`、`Function` 両方のコンストラクタが試行された場合、エラーをスローします。たとえば、`const sum = new Function('a', 'b', 'return a + b')` はエラーをスローします。

タイマー

`setTimeout()`、`setImmediate()`、`clearTimeout()` 関数はサポートされていません。関数実行中に defer または yield する規定はありません。関数は同期的に実行しないと完了できません。

日付とタイムスタンプ

セキュリティ上の理由から、高解像度タイマーにはアクセスできません。現在の時刻を照会するすべての `Date` メソッドは、単一の関数実行の存続期間中は常に同じ値を返します。返されるタイムスタンプは、関数の実行を開始した時刻です。したがって、関数内で経過時間を測定することはできません。

ファイルシステムへのアクセス

ファイルシステムにはアクセスできません。

ネットワークアクセス

ネットワークコールはサポートされていません。たとえば、XHR、HTTP (S)、ソケットはサポートされていません。

キー値ストアのヘルパーメソッド

このセクションは、CloudFront キー値ストアを使用して作成する関数にキー値を含める場合に適用されます。CloudFront Functions には、キー値ストアから値を読み取る 3 つのヘルパーメソッドを提供するモジュールがあります。

このモジュールを関数コードで使用するには、関数にキー値ストアを関連付ける必要があります。

次に、関数コードの最初の行に、以下のステートメントを含めます。

```
import cf from 'cloudfront';
const kvsId = "key value store ID";
const kvsHandle = cf.kvs(kvsId);
```

`###### ID` は a1b2c3d4-5678-90ab-cdef-EXAMPLE1 のようになります。

get() 方法

このメソッドを使用して、指定したキー名のキー値を返します。

リクエスト

```
get("key", options);
```

- key: 値をフェッチする必要があるキーの名前
- options: 1 つのオプション format があります。これにより、関数はデータを正しく解析します。使用できる値:
  - string: (デフォルト) UTF8 エンコード
  - json
  - bytes: 未加工のバイナリデータバッファ

リクエストの例

```
const value = await kvsHandle.get("myFunctionKey", { format: "string"});
```

レスポンス

レスポンスは promise であり、options を使用してリクエストした形式の値に解決されます。デフォルトでは、値は文字列として返されます。

exists() 方法

このメソッドを使用して、キーがキー値ストアに存在するかどうかを確認します。

リクエスト

```
exists("key");
```

リクエストの例

```
const exist = await kvsHandle.exists("myFunctionkey");
```

レスポンス

レスポンスは promise であり、ブール値 (true または false) を返します。この値は、キーがキー値ストアに存在するかどうかを示します。

エラー処理

リクエストしたキーが、関連するキー値ストアに存在しない場合、get() メソッドはエラーを返します。このユースケースを管理するには、コードに try および catch ブロックを追加できます。

meta() 方法

このメソッドを使用して、キー値ストアに関するメタデータを返します。

リクエスト

```
meta();
```

リクエストの例

```
const meta = await kvsHandle.meta();
```

レスポンス

レスポンスは promise で、以下のプロパティを持つオブジェクトに解決されます。

- creationDateTime: キー値ストアが作成された ISO 8601 形式の日付と時刻。
- lastUpdatedDateTime: キー値ストアがソースから最後に同期された ISO 8601 形式の日付と時刻。値にはエッジへの伝達時間は含まれていません。
- keyCount: ソースからの最後の同期後の KVS 内のキーの合計数。

レスポンスの例

```
{keyCount:3,creationDateTime:2023-11-30T23:07:55.765Z,lastUpdatedDateTime:2023-12-15T03:57:52.4
```

CloudFront Functions のコード例

CloudFront Functions の関数コードの記述を簡単に開始するには、以下の例を参考にしてください。これらの例は、GitHub の「amazon-cloudfront-functions リポジトリ」にもあります。

トピック

- レスポンスに Cache-Control ヘッダーを追加する
- Cross-Origin Resource Sharing (CORS) ヘッダーをレスポンスに追加
- Cross-Origin Resource Sharing (CORS) ヘッダーをリクエストに追加
- レスポンスにセキュリティヘッダーを追加する
- リクエストに True-Client-IP ヘッダーを追加する
- ビューワーを新しい URL にリダイレクトさせる
- index.html を追加してファイル名を含まない URL をリクエストする
- リクエストの単純なトークンを検証する
- async および await を使用します。
- クエリ文字列パラメータの正規化
- 関数でキーと値のペアを使用する

レスポンスに Cache-Control ヘッダーを追加する

次のビューワーレスポンス関数は、レスポンスに `Cache-Control` HTTP ヘッダーを追加します。ヘッダーは `max-age` ディレクティブを使用して、最大 2 年 (63,072,000 秒) の応答をキャッシュするようにウェブブラウザに指示します。詳細については、MDN Web Docs の Web サイトの「Cache-Control」を参照してください。

この例を GitHub で見てみましょう。

JavaScript runtime 2.0

```
async function handler(event) {
    const response = event.response;
    const headers = response.headers;

    // Set the cache-control header
     headers['cache-control'] = {value: 'public, max-age=63072000'};

    // Return response to viewers
```

```
    return response;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var response = event.response;
    var headers = response.headers;

    // Set the cache-control header
    headers['cache-control'] = {value: 'public, max-age=63072000'};

    // Return response to viewers
    return response;
}
```

Cross-Origin Resource Sharing (CORS) ヘッダーをレスポンスに追加

次のビューワーレスポンス関数は、レスポンスに `Access-Control-Allow-Origin` HTTP ヘッダーを追加します (まだ追加されていない場合)。このヘッダーは、Cross-Origin Resource Sharing (CORS) の一部です。ヘッダーの値 (*) は、任意のオリジンからのコードがこのリソースにアクセスできるように Web ブラウザに指示します。詳細については、MDN Web Docs Web サイトの「Access-Control-Allow-Origin」を参照してください。

この例を GitHub で見てみましょう。

JavaScript runtime 2.0

```
async function handler(event)  {
    const request = event.request;
    const response  = event.response;

    // If Access-Control-Allow-Origin CORS header is missing, add it.
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation.
    if (!response.headers['access-control-allow-origin'] &&
 request.headers['origin']) {
        response.headers['access-control-allow-origin'] = {value:
 request.headers['origin'].value};
        console.log("Access-Control-Allow-Origin was missing, adding it now.");
    }
```

```
    return response;
}
```

JavaScript runtime 1.0

```
function handler(event)  {
    var response  = event.response;
    var headers  = response.headers;

    // If Access-Control-Allow-Origin CORS header is missing, add it.
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation.
    if (!headers['access-control-allow-origin']) {
        headers['access-control-allow-origin'] = {value: "*"};
        console.log("Access-Control-Allow-Origin was missing, adding it now.");
    }

    return response;
}
```

Cross-Origin Resource Sharing (CORS) ヘッダーをリクエストに追加

次のビューワーリクエスト関数は、リクエストに `Origin` HTTP ヘッダーを追加します (まだ追加されていない場合)。このヘッダーは、Cross-Origin Resource Sharing (CORS) の一部です。この例では、ヘッダーの値をリクエストの `Host` ヘッダーの値に設定します。詳細については、MDN Web Docs の Web サイトの「 Origin」を参照してください。

この例を GitHub で見てみましょう。

JavaScript runtime 2.0

```
async function handler(event) {
    const request = event.request;
    const headers = request.headers;
    const host = request.headers.host.value;

   // If origin header is missing, set it equal to the host header.
   if (!headers.origin)
       headers.origin = {value:`https://${host}`};

   return request;
```

```
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var request = event.request;
    var headers = request.headers;
    var host = request.headers.host.value;

   // If origin header is missing, set it equal to the host header.
   if (!headers.origin)
       headers.origin = {value:`https://${host}`};

   return request;
}
```

レスポンスにセキュリティヘッダーを追加する

次のビューワーレスポンス関数は、いくつかの一般的なセキュリティ関連の HTTP ヘッダーをレスポンスに追加します。詳細については、MDN Web Docs Web サイトの以下のページを参照してください。

- Strict-Transport-Security
- Content-Security-Policy
- X-Content-Security-Policy
- X-Frame-Options
- X-XSS-Protection

この例を GitHub で見てみましょう。

JavaScript runtime 2.0

```
async function handler(event) {
    const response = event.response;
    const headers = response.headers;

    // Set HTTP security headers
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation
```

```
    headers['strict-transport-security'] = { value: 'max-age=63072000;
 includeSubdomains; preload'};
    headers['content-security-policy'] = { value: "default-src 'none'; img-src
 'self'; script-src 'self'; style-src 'self'; object-src 'none'; frame-ancestors
 'none'"};
    headers['x-content-type-options'] = { value: 'nosniff'};
    headers['x-frame-options'] = {value: 'DENY'};
    headers['x-xss-protection'] = {value: '1; mode=block'};
    headers['referrer-policy'] = {value: 'same-origin'};

    // Return the response to viewers
    return response;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var response = event.response;
    var headers = response.headers;

    // Set HTTP security headers
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation
    headers['strict-transport-security'] = { value: 'max-age=63072000;
 includeSubdomains; preload'};
    headers['content-security-policy'] = { value: "default-src 'none'; img-src
 'self'; script-src 'self'; style-src 'self'; object-src 'none'"};
    headers['x-content-type-options'] = { value: 'nosniff'};
    headers['x-frame-options'] = {value: 'DENY'};
    headers['x-xss-protection'] = {value: '1; mode=block'};

    // Return the response to viewers
    return response;
}
```

リクエストに True-Client-IP ヘッダーを追加する

次のビューワーリクエスト関数は、ビューワーの IP アドレスをヘッダーの値として、`True-Client-IP` HTTP ヘッダーをリクエストに追加します。CloudFront がオリジンにリクエストを送信すると、オリジンはリクエスト送信元である CloudFront ホストの IP アドレスは特定できますが、リクエストを最初に CloudFront に送信したビューワー (クライアント) の IP アドレスは特定できませ

ん。この関数は、True-Client-IP ヘッダーを追加してオリジンがビューワーの IP アドレスを確認できるようにします。

> **Important**
>
> CloudFront がこのヘッダーをオリジンリクエストに含めるようにするには、オリジンリクエストポリシーの許可ヘッダーリストにそのヘッダーを追加する必要があります 。

この例を GitHub で見てみましょう。

JavaScript runtime 2.0

```
async function handler(event) {
    var request = event.request;
    var clientIP = event.viewer.ip;

    //Add the true-client-ip header to the incoming request
    request.headers['true-client-ip'] = {value: clientIP};

    return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var request = event.request;
    var clientIP = event.viewer.ip;

    //Add the true-client-ip header to the incoming request
    request.headers['true-client-ip'] = {value: clientIP};

    return request;
}
```

ビューワーを新しい URL にリダイレクトさせる

次のビューワーリクエスト関数は、リクエストが特定の国から送信されたときに、ビューワーを国固有の URL にリダイレクトするためのレスポンスを生成します。この関数は、CloudFront-Viewer-Country ヘッダーの値に依存して、ビューワーの国を判断します。

> **Important**
>
> この機能を使用するには、キャッシュポリシーまたはオリジンリクエストポリシーの許可ヘッダーリストに CloudFront-Viewer-Country ヘッダーを追加し、このヘッダーが受信リクエストにも追加されるように CloudFront を設定する必要があります。

この例では、ビューワーリクエストがドイツから送信された場合、ビューワーをドイツ固有の URL にリダイレクトします。ビューワーリクエストがドイツから来ていない場合、この関数は元の変更されていないリクエストを返します。

この例を GitHub で見てみましょう。

JavaScript runtime 2.0

```
async function handler(event) {
    const request = event.request;
    const headers = request.headers;
    const host = request.headers.host.value;
    const country = Symbol.for('DE'); // Choose a country code
    const newurl = `https://${host}/de/index.html`; // Change the redirect URL to
 your choice

    if (headers['cloudfront-viewer-country']) {
        const countryCode = Symbol.for(headers['cloudfront-viewer-country'].value);
        if (countryCode === country) {
            const response = {
                statusCode: 302,
                statusDescription: 'Found',
                headers:
                    { "location": { "value": newurl } }
                }

            return response;
        }
    }
    return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
```

```
    var request = event.request;
    var headers = request.headers;
    var host = request.headers.host.value;
    var country = 'DE' // Choose a country code
    var newurl = `https://${host}/de/index.html` // Change the redirect URL to your
 choice

    if (headers['cloudfront-viewer-country']) {
        var countryCode = headers['cloudfront-viewer-country'].value;
        if (countryCode === country) {
            var response = {
                statusCode: 302,
                statusDescription: 'Found',
                headers:
                    { "location": { "value": newurl } }
                }

            return response;
        }
    }
    return request;
}
```

リライトとリダイレクトの詳細については、「AWS workshop studio」の「エッジ関数を使用したリライトとリダイレクトの処理」を参照してください。

index.html を追加してファイル名を含まない URL をリクエストする

次のビューワーリクエスト関数は、URL にファイル名や拡張子を含まないリクエストに `index.html` を付加します。この機能は、単一ページアプリケーションや Amazon S3 バケットでホストされている静的に生成されたウェブサイトに便利です。

この例を GitHub で見てみましょう。

JavaScript runtime 2.0

```
async function handler(event) {
    const request = event.request;
    const uri = request.uri;

    // Check whether the URI is missing a file name.
    if (uri.endsWith('/')) {
```

```
        request.uri += 'index.html';
    }
    // Check whether the URI is missing a file extension.
    else if (!uri.includes('.')) {
        request.uri += '/index.html';
    }

    return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var request = event.request;
    var uri = request.uri;

    // Check whether the URI is missing a file name.
    if (uri.endsWith('/')) {
        request.uri += 'index.html';
    }
    // Check whether the URI is missing a file extension.
    else if (!uri.includes('.')) {
        request.uri += '/index.html';
    }

    return request;
}
```

リクエストの単純なトークンを検証する

次のビューアーリクエスト関数は、リクエストのクエリ文字列の JSON ウェブトークン (JWT) を検証します。トークンが有効な場合、関数は元の変更されていないリクエストを CloudFront に返します。トークンが有効でない場合、関数はエラーレスポンスを生成します。この関数は `crypto` モジュールを使用します。詳細については、「ビルトインモジュール」を参照してください。

この関数は、リクエストの `jwt` という名前のクエリ文字列パラメータに JWT 値が含まれていることを前提としています。

> **Warning**
>
> この関数を使用するには、関数コードにシークレットキーを配置する必要があります。

この例を GitHub で見てみましょう。

JavaScript runtime 2.0

```
const crypto = require('crypto');

//Response when JWT is not valid.
const response401 = {
    statusCode: 401,
    statusDescription: 'Unauthorized'
};

function jwt_decode(token, key, noVerify, algorithm) {
    // check token
    if (!token) {
        throw new Error('No token supplied');
    }
    // check segments
    const segments = token.split('.');
    if (segments.length !== 3) {
        throw new Error('Not enough or too many segments');
    }

    // All segment should be base64
    const headerSeg = segments[0];
    const payloadSeg = segments[1];
    const signatureSeg = segments[2];

    // base64 decode and parse JSON
    const header = JSON.parse(_base64urlDecode(headerSeg));
    const payload = JSON.parse(_base64urlDecode(payloadSeg));

    if (!noVerify) {
        const signingMethod = 'sha256';
        const signingType = 'hmac';

        // Verify signature. `sign` will return base64 string.
        const signingInput = [headerSeg, payloadSeg].join('.');

        if (!_verify(signingInput, key, signingMethod, signingType, signatureSeg)) {
            throw new Error('Signature verification failed');
        }

        // Support for nbf and exp claims.
```

```
        // According to the RFC, they should be in seconds.
        if (payload.nbf && Date.now() < payload.nbf*1000) {
            throw new Error('Token not yet active');
        }

        if (payload.exp && Date.now() > payload.exp*1000) {
            throw new Error('Token expired');
        }
    }

    return payload;
}

//Function to ensure a constant time comparison to prevent
//timing side channels.
function _constantTimeEquals(a, b) {
    if (a.length != b.length) {
        return false;
    }

    var xor = 0;
    for (var i = 0; i < a.length; i++) {
    xor |= (a.charCodeAt(i) ^ b.charCodeAt(i));
    }

    return 0 === xor;
}

function _verify(input, key, method, type, signature) {
    if(type === "hmac") {
        return _constantTimeEquals(signature, _sign(input, key, method));
    }
    else {
        throw new Error('Algorithm type not recognized');
    }
}

function _sign(input, key, method) {
    return crypto.createHmac(method, key).update(input).digest('base64url');
}

function _base64urlDecode(str) {
    return Buffer.from(str, 'base64url')
}
```

```
function handler(event) {
    const request = event.request;
    //Secret key used to verify JWT token.
    //Update with your own key.
    var key = "LzdWGpAToQ1DqYuzHxE6YOqi7G3X2yvNBot9mCXfx5k";

    // If no JWT token, then generate HTTP redirect 401 response.
    if(!request.querystring.jwt) {
        console.log("Error: No JWT in the querystring");
        return response401;
    }

    const jwtToken = request.querystring.jwt.value;

    try{
        jwt_decode(jwtToken, key);
    }
    catch(e) {
        console.log(e);
        return response401;
    }

    //Remove the JWT from the query string if valid and return.
    delete request.querystring.jwt;
    console.log("Valid JWT token");
    return request;
}
```

JavaScript runtime 1.0

```
var crypto = require('crypto');

//Response when JWT is not valid.
var response401 = {
    statusCode: 401,
    statusDescription: 'Unauthorized'
};

function jwt_decode(token, key, noVerify, algorithm) {
    // check token
    if (!token) {
        throw new Error('No token supplied');
```

```
    }
    // check segments
    var segments = token.split('.');
    if (segments.length !== 3) {
        throw new Error('Not enough or too many segments');
    }

    // All segment should be base64
    var headerSeg = segments[0];
    var payloadSeg = segments[1];
    var signatureSeg = segments[2];

    // base64 decode and parse JSON
    var header = JSON.parse(_base64urlDecode(headerSeg));
    var payload = JSON.parse(_base64urlDecode(payloadSeg));

    if (!noVerify) {
        var signingMethod = 'sha256';
        var signingType = 'hmac';

        // Verify signature. `sign` will return base64 string.
        var signingInput = [headerSeg, payloadSeg].join('.');

        if (!_verify(signingInput, key, signingMethod, signingType, signatureSeg)) {
            throw new Error('Signature verification failed');
        }

        // Support for nbf and exp claims.
        // According to the RFC, they should be in seconds.
        if (payload.nbf && Date.now() < payload.nbf*1000) {
            throw new Error('Token not yet active');
        }

        if (payload.exp && Date.now() > payload.exp*1000) {
            throw new Error('Token expired');
        }
    }

    return payload;
}

function _verify(input, key, method, type, signature) {
    if(type === "hmac") {
        return (signature === _sign(input, key, method));
```

```
    }
    else {
        throw new Error('Algorithm type not recognized');
    }
}

function _sign(input, key, method) {
    return crypto.createHmac(method, key).update(input).digest('base64url');
}

function _base64urlDecode(str) {
    return String.bytesFrom(str, 'base64url')
}

function handler(event) {
    var request = event.request;

    //Secret key used to verify JWT token.
    //Update with your own key.
    var key = "LzdWGpAToQ1DqYuzHxE6YOqi7G3X2yvNBot9mCXfx5k";

    // If no JWT token, then generate HTTP redirect 401 response.
    if(!request.querystring.jwt) {
        console.log("Error: No JWT in the querystring");
        return response401;
    }

    var jwtToken = request.querystring.jwt.value;

    try{
        jwt_decode(jwtToken, key);
    }
    catch(e) {
        console.log(e);
        return response401;
    }

    //Remove the JWT from the query string if valid and return.
    delete request.querystring.jwt;
    console.log("Valid JWT token");
    return request;
}
```

async および await を使用します。

CloudFront Functions JavaScript ランタイム関数 2.0 には、Promise オブジェクトを処理するための async および await 構文が用意されています。Promise は遅延した結果を表し、async とマークされた関数のキーワード await を使用してアクセスできます。さまざまな新しい WebCrypto 関数が Promise を使用しています。

Promise オブジェクトの詳細については、「Promise」を参照してください。

> **Note**
>
> 次のコードサンプルでは、JavaScript ランタイム 2.0 を使用する必要があります。

```
async function answer() {
    return 42;
}

// Note: async, await can be used only inside an async function.

async function handler(event) {
    // var answer_value = answer(); // returns Promise, not a 42 value
    let answer_value = await answer(); // resolves Promise, 42
    console.log("Answer"+answer_value);
    event.request.headers['answer'] = { value : ""+answer_value };
    return event.request;
}
```

次の JavaScript コード例は、then チェーンメソッドを使用して Promise を表示する方法を示しています。catch を使用してエラーを表示できます。

```
async function answer() {
    return 42;
}

async function squared_answer() {
   return answer().then(value => value * value)
}
// note async, await can be used only inside async function
async function handler(event) {
    // var answer_value = answer(); // returns Promise, not a 42 value
```

```
    let answer_value = await squared_answer(); // resolves Promise, 42
    console.log("Answer"+answer_value);
    event.request.headers['answer'] = { value : ""+answer_value };
    return event.request;
}
```

クエリ文字列パラメータの正規化

クエリ文字列パラメータを正規化して、キャッシュヒット率を高めることができます。

次の例は、JavaScript ランタイム 1.0 および 2.0 で動作します。次の例は、CloudFront がリクエストをオリジンに転送する前に、クエリ文字列をアルファベット順に並べることで、キャッシュヒット率を高める方法を示しています。

```
function handler(event) {
     var qs=[];
     for (var key in event.request.querystring) {
         if (event.request.querystring[key].multiValue) {
             event.request.querystring[key].multiValue.forEach((mv) => {qs.push(key +
 "=" + mv.value)});
         } else {
             qs.push(key + "=" + event.request.querystring[key].value);
         }
     };

     event.request.querystring = qs.sort().join('&');


     return event.request;
}
```

関数でキーと値のペアを使用する

キー値ストアのキーと値のペアを関数で使用できます。

> **Note**
>
> 次のコードサンプルでは JavaScript ランタイム 2.0 を使用する必要があります。

この例は、HTTP リクエスト内の URL の内容を使用してキー値ストア内のカスタムパスを検索する関数を示しています。次に、CloudFront はそのカスタムパスを使用してリクエストを行います。この関数は、ウェブサイトに含まれる複数のパスを管理するのに役立ちます。

```
import cf from 'cloudfront';

// Declare the ID of the key value store that you have associated with this function
// The import fails at runtime if the specified key value store is not associated with
 the function

const kvsId = "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111";

const kvsHandle = cf.kvs(kvsId);

async function handler(event) {
    const request = event.request;
    // Use the first segment of the pathname as key
    // For example http(s)://domain/<key>/something/else
    const pathSegments = request.uri.split('/')
    const key = pathSegments[1]
    try {
        // Replace the first path of the pathname with the value of the key
        // For example http(s)://domain/<value>/something/else
        pathSegments[1] = await kvsHandle.get(key);
        const newUri = pathSegments.join('/');
        console.log(`${request.uri} -> ${newUri}`)
        request.uri = newUri;
    } catch (err) {
        // No change to the pathname if the key is not found
        console.log(`${request.uri} | ${err}`);
    }
    return request;
}
```

関数を作成する

関数は 2 段階で作成します。

1. 関数コードを JavaScript として作成します。CloudFront コンソールのデフォルトの例を使用することも、独自に記述することもできます。詳細については、次のトピックを参照してください。
   - 関数コードを記述する

- the section called “イベントの構造”
- CloudFront Functions のコード例

2. CloudFront を使用して関数を作成し、コードを含めます。コードは関数内にあります (リファレンスとしてではありません)。

Console

関数を作成するには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home#/functions) にサインインし、[関数] ページを選択します。
2. [Create function (関数の作成)] を選択します。
3. AWS アカウント内での一意な関数名を入力し、Java Script バージョンを選択して、[続行] を選択します。新しい関数の詳細ページが表示されます。

> **Note**
>
> 関数で [キーと値のペア] を使用する場合は、Java Script ランタイム 2.0 を選択する必要があります。

4. [関数コード] セクションで、[ビルド] タブを選択し、関数コードを入力します。[ビルド] タブに含まれるサンプルコードは、関数コードの基本的な構文を示しています。
5. [Save changes] (変更の保存) をクリックします。
6. 関数コードでキーと値のペアを使用する場合は、キー値ストアを関連付ける必要があります。

   キー値ストアは、関数の作成時に関連付けることができます。または、後で関数を更新して関連付けることもできます。

   キーと値のストアを今すぐ関連付けるには、次の手順に従います。

   - [KeyValueStore を関連付け] セクションに移動して、[既存の KeyValueStore を関連付け] を選択します。
   - 関数のキーと値のペアを含むキー値ストアを選択し、[KeyValueStore を関連付ける] を選択します。

   CloudFront はストアを直ちに関数に関連付けます。関数を保存する必要はありません。

CLI

CLI を使用する場合、通常は最初に関数コードをファイルに作成し、次に AWS CLI を使用して関数を作成します。

関数を作成するには

1. 関数コードをファイルに作成し、コンピュータが接続できるディレクトリに保存します。
2. 次の例に示すようにコマンドを実行します。この例では、`fileb://` 表記を使用してファイルを渡します。コマンドを読みやすくするために改行も含まれています。

```
aws cloudfront create-function \
    --name MaxAge \
    --function-config '{"Comment":"Max Age 2 years","Runtime":"cloudfront-
js-2.0","KeyValueStoreAssociations":{"Quantity":1,"Items":
[{"KeyValueStoreARN":"arn:aws:cloudfront::111122223333:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"}]}}' \
    --function-code fileb://function-max-age-v1.js
```

> **メモ**
>
> - Runtime – Java Script のバージョン。関数でキーと値のペアを使用するには、バージョン 2.0 を指定する必要があります。
> - KeyValueStoreAssociations – 関数がキーと値のペアを使用する場合、関数の初回作成時にキー値ストアを関連付けることができます。または、update-function を使用して後で関連付けることもできます。各関数に関連付けることができるキー値ストアは 1 つだけのため、Quantity は常に 1 です。

コマンドが成功した場合は、以下のような出力が表示されます。

```
ETag: ETVABCEXAMPLE
FunctionSummary:
  FunctionConfig:
    Comment: Max Age 2 years
    Runtime: cloudfront-js-2.0
    KeyValueStoreAssociations= \
      {Quantity=1, \
```

```
      Items=[{KeyValueStoreARN='arn:aws:cloudfront::111122223333:key-value-
store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]} \
  FunctionMetadata:
    CreatedTime: '2021-04-18T20:38:56.915000+00:00'
    FunctionARN: arn:aws:cloudfront::111122223333:function/MaxAge
    LastModifiedTime: '2023-11-19T20:38:56.915000+00:00'
    Stage: DEVELOPMENT
  Name: MaxAge
  Status: UNPUBLISHED
Location: https://cloudfront.amazonaws.com/2020-05-31/function/
arn:aws:cloudfront:::function/MaxAge
```

ほとんどの情報はリクエストから繰り返されます。その他の情報は CloudFront によって追加されます。

> **メモ**
>
> - ETag – この値は、キー値ストアを変更するたびに変わります。この値と関数名を使用して、以後、この関数を参照します。必ず現在の ETag を使用してください。
> - FunctionARN – CloudFront 関数の ARN。
> - 111122223333 – AWS アカウント。
> - Stage – 関数のステージ (LIVE または DEVELOPMENT) 。
> - Status – 関数のステータス (PUBLISHED または UNPUBLISHED)。

関数を作成すると、DEVELOPMENT ステージに追加されます。関数は、テストしてから発行することをお勧めします。関数を発行すると、関数のステージは LIVE に変わります。

関数をテストする

関数をライブステージ (本番環境) にデプロイする前に、テストして期待どおりに動作することを確認できます。関数をテストするには、CloudFront ディストリビューションが本番環境で受信する可能性がある HTTP リクエストやレスポンスを表すイベントオブジェクトを指定します。

CloudFront Functions は以下の処理を行います。

1. 指定されたイベントオブジェクトを入力として使用して、関数を実行します。

2. 関数の結果 (変更されたイベントオブジェクト) を、関数ログまたはエラーメッセージ、および関数のコンピューティング使用率とともに返します。コンピューティング使用率の詳細については、「the section called “コンピューティング使用率を理解する”」を参照してください。

目次

イベントオブジェクトをセットアップする

関数をテストする前に、テストに使用するイベントオブジェクトをセットアップする必要があります。これには複数のオプションがあります。

オプション 1: イベントオブジェクトを保存せずにセットアップする

CloudFront コンソールのビジュアルエディターでイベントオブジェクトをセットアップしても、保存はできません。

このイベントオブジェクトを使用して、保存されていなくても CloudFront コンソールから関数をテストできます。

オプション 2: ビジュアルエディターでイベントオブジェクトを作成する

CloudFront コンソールのビジュアルエディターでイベントオブジェクトをセットアップしても、保存はできません。関数ごとに 10 個のイベントオブジェクトを作成して、例えば、考えられるさまざまな入力をテストできます。

この方法でイベントオブジェクトを作成すると、イベントオブジェクトを使用して CloudFront コンソールで関数をテストできます。AWS API や SDK を使用して関数をテストする場合には使用できません。

オプション 3: テキストエディターを使用してイベントオブジェクトを作成する

テキストエディターを使用して、イベントオブジェクトを JSON 形式で作成できます。イベントオブジェクトの構造については、「イベントの構造」を参照してください。

このイベントオブジェクトを使用して、CLI で関数をテストできます。ただし、これを使用して CloudFront コンソールで関数をテストすることはできません。

イベントオブジェクトを作成するには (オプション 1 または 2)

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home#/functions) にサインインし、[関数] ページを選択します。

   テストする関数を選択します。
2. 関数の詳細ページで、[テスト] タブを選択します。
3. [イベントタイプ] で、以下のいずれかのオプションを選択します。

   - 関数が HTTP リクエストを変更したり、リクエストに基づいてレスポンスを生成する場合は、[ビューワーリクエスト] を選択します。[リクエスト] セクションが表示されます。
   - [ビューアーレスポンス] を選択します。[リクエスト] セクションと [レスポンス] セクションが表示されます。
4. イベントに含めるフィールドに入力します。[JSON を編集] を選択すると、未加工の JSON を表示できます。
5. (オプション) イベントを保存するには、[保存] を選択し、[テストイベントを保存] に名前を入力して [保存] を選択します。

   [JSON を編集] を選択し、未加工の JSON をコピーして、CloudFront 外の独自のファイルに保存することもできます。

イベントオブジェクトを作成するには (オプション 3)

テキストエディタを使用してイベントオブジェクトを作成します。このファイルは、コンピューターが接続できるディレクトリに保存します。

以下のガイドラインに確実に従ってください。

- `distributionDomainName`、`distributionId`、`requestId` の各フィールドは省略します。
- ヘッダー、Cookie、クエリ文字列の名前は必ず小文字にします。

この方法でイベントオブジェクトを作成する方法の 1 つは、ビジュアルエディターを使用してサンプルを作成することです。サンプルが正しいフォーマットになっていることを確認できます。そして、未加工の JSON をコピーし、テキストエディターに貼り付け、ファイルを保存することができます。

イベントの構造の詳細については、「イベントの構造」を参照してください。

関数をテストする

関数は、CloudFront コンソールまたは AWS Command Line Interface (AWS CLI) を使用してテストできます。

Console

関数をテストするには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home#/functions) にサインインし、[関数] ページを選択します。
2. テストする関数を選択します。
3. [テスト] タブを選択します。
4. 正しいイベントが表示されていることを確認します。現在表示されているイベントから切り替えるには、[テストイベントを選択] フィールドで別のイベントを選択します。
5. [関数をテスト] を選択します。コンソールに、関数ログとコンピューティング使用率を含む、関数の出力が表示されます。

CLI

aws cloudfront test-function コマンドを使用して関数をテストできます。

関数をテストするには

1. コマンドラインウィンドウを開きます。
2. 指定したファイルがある同じディレクトリから、次のコマンドを実行します。

   この例では、`fileb://` 表記を使用してイベントオブジェクトファイルを渡します。コマンドを読みやすくするために改行も含まれています。

```
aws cloudfront test-function \
    --name MaxAge \
    --if-match ETVABCEXAMPLE \
    --event-object fileb://event-maxage-test01.json \
    --stage DEVELOPMENT
```

メモ

- 関数は名前と ETag (`if-match` パラメータ内) で参照します。イベントオブジェクトはファイルシステム内の口ケーションによって参照します。
- ステージは、`DEVELOPMENT` または `LIVE` の場合があります。

コマンドが成功した場合は、以下のような出力が表示されます。

```
TestResult:
  ComputeUtilization: '21'
  FunctionErrorMessage: ''
  FunctionExecutionLogs: []
  FunctionOutput: '{"response":{"headers":{"cloudfront-functions":
{"value":"generated-by-CloudFront-Functions"},"location":{"value":"https://
aws.amazon.com/cloudfront/"}},"statusDescription":"Found","cookies":
{},"statusCode":302}}'
  FunctionSummary:
    FunctionConfig:
      Comment: MaxAge function
      Runtime: cloudfront-js-2.0
      KeyValueStoreAssociations= \
      {Quantity=1, \
      Items=[{KeyValueStoreARN='arn:aws:cloudfront::111122223333:key-value-
store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]} \
    FunctionMetadata:
      CreatedTime: '2021-04-18T20:38:56.915000+00:00'
      FunctionARN: arn:aws:cloudfront::111122223333:function/MaxAge
      LastModifiedTime: '2023-17-20T10:38:57.057000+00:00'
      Stage: DEVELOPMENT
    Name: MaxAge
    Status: UNPUBLISHED
```

メモ

- `FunctionExecutionLogs` には、関数が `console.log()` ステートメントに書き込んだ (該当する場合) ログ行のリストが含まれます。

- `ComputeUtilization` には、関数の実行に関する情報が含まれています。「the section called “コンピューティング使用率を理解する”」を参照してください。
- `FunctionOutput` には、関数が返したイベントオブジェクトが含まれます。

コンピューティング使用率を理解する

コンピューティング使用率は、関数の実行にかかった時間 (最大許容時間に対するパーセンテージ) です。たとえば、値 35 は、関数が最大許容時間の 35% で完了したことを意味します。

関数が最大許容時間を継続的に超える場合、CloudFront で関数がスロットリングされます。次のリストは、コンピューティング使用率の値に基づいて関数がスロットリングされる可能性を説明しています。

コンピューティング使用率値:

- 1 ~ 50 — 関数は最大許容時間を十分に下回っており、スロットリングなしで実行する必要があります。
- 51 ~ 70 — 関数が最大許容時間に近づいています。関数コードの最適化を検討してください。
- 71 ~ 100 — 関数が最大許容時間に非常に近いか、それを超えています。ディストリビューションに関連付けると、CloudFront でこの関数が抑制される可能性があります。

関数を更新する

関数はいつでも更新できます。変更は、`DEVELOPMENT` ステージ内の関数のバージョンに対してのみ行われます。更新を `DEVELOPMENT` ステージから `LIVE` にコピーするには、関数を発行する必要があります。

関数のコードを更新するには、CloudFront コンソールまたは AWS Command Line Interface (AWS CLI) を使用できます。

Console

関数コードを更新するには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home#/functions) にサインインし、[関数] ページを選択します。

   対象の関数を選択します。

2. [編集] を選択し、以下の変更を行います。

   - [詳細] セクションのフィールドを更新します。
   - 関連するキー値ストアを変更または削除します。キー値ストアの詳細については、「the section called “CloudFront KeyValueStore の使用”」を参照してください。
   - 関数コードを変更します。[ビルド] タブを選択し、変更を加え、[変更を保存] を選択してコードへの変更を保存します。

CLI

関数コードを更新するには

1. コマンドラインウィンドウを開きます。
2. 以下のコマンドを実行します。

   この例では、`fileb://` 表記を使用してファイルを渡します。コマンドを読みやすくするために改行も含まれています。

```
aws cloudfront update-function \
    --name MaxAge \
    --function-config '{"Comment":"Max Age 2 years","Runtime":"cloudfront-
js-2.0","KeyValueStoreAssociations":{"Quantity":1,"Items":
[{"KeyValueStoreARN":"arn:aws:cloudfront::111122223333:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"}]}}' \
    --function-code fileb://function-max-age-v1.js \
    --if-match ETVABCEXAMPLE
```

> **メモ**
>
> - 関数は名前と ETag (`if-match` パラメータ内) の両方で識別します。必ず現在の ETag を使用してください。describe オペレーションを使用して取得できます。
> - 変更したくない場合でも、`function-code` を含める必要があります。
> - `function-config` には注意してください。設定に保持するすべてのものを渡す必要があります。具体的には、キー値ストアを次のように処理してください。

- 既存のキー値ストアの関連付け (存在する場合) を維持するには、既存のストアの名前を指定します。
- 関連付けを変更するには、新しいキー値ストアの名前を指定します。
- 関連付けを削除するには、KeyValueStoreAssociations パラメータを省略します。

コマンドが成功した場合は、以下のような出力が表示されます。

```
ETag: ETVXYZEXAMPLE
FunctionSummary:
  FunctionConfig:
    Comment: Max Age 2 years \
    Runtime: cloudfront-js-2.0 \
    KeyValueStoreAssociations= \
      {Quantity=1, \
      Items=[{KeyValueStoreARN='arn:aws:cloudfront::111122223333:key-value-
store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]} \
  FunctionMetadata: \
    CreatedTime: '2021-04-18T20:38:56.915000+00:00' \
    FunctionARN: arn:aws:cloudfront::111122223333:function/MaxAge \
    LastModifiedTime: '2023-12-19T23:41:15.389000+00:00' \
    Stage: DEVELOPMENT \
  Name: MaxAge \
  Status: UNPUBLISHED
```

ほとんどの情報はリクエストから繰り返されます。その他の情報は CloudFront によって追加されます。

メモ

- ETag – この値は、キー値ストアを変更するたびに変わります。
- FunctionARN – CloudFront 関数の ARN。
- Stage – 関数のステージ (LIVE または DEVELOPMENT) 。
- Status – 関数のステータス (PUBLISHED または UNPUBLISHED)。

関数を発行する

関数を発行すると、関数が DEVELOPMENT ステージから LIVE ステージにコピーされます。

キャッシュ動作が関連付けられていない関数を発行した場合は、関数をキャッシュ動作に関連付けることができます。キャッシュ動作は、LIVE ステージの関数にのみ関連付けることができます。

> **Important**
>
> - 関数を発行する前に、テストすることをお勧めします。
> - 関数を発行すると、関数に関連付けられたすべてのキャッシュ動作は (ディストリビューションのデプロイが完了しだい)、新しく発行したコピーを自動的に使用するようになります。

CloudFront コンソールまたは AWS CLI を使用して、関数を公開できます。

Console

関数を発行するには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home#/functions) にサインインし、[関数] ページを選択します。
2. 対象の関数を選択します。
3. [発行] タブを選択し、[発行] を選択します。関数が 1 つ以上のキャッシュ動作に既にアタッチされている場合は、[発行して更新] を選択します。
4. (オプション) 関数に関連付けられたディストリビューションを表示するには、[Associated CloudFront distributions] を選択して、そのセクションを展開します。

正常に完了すると、ページの上部に「**###**は正常に発行されました」というバナーが表示されます。[Build] タブから [Live] を選択して、関数コードのライブバージョンを表示することもできます。

CLI

関数を発行するには

1. コマンドラインウィンドウを開きます。

2. 次の aws cloudfront publish-function コマンドを実行します。この例では、例を読みやすくするために改行されています。

```
aws cloudfront publish-function \
    --name MaxAge \
    --if-match ETVXYZEXAMPLE
```

コマンドが成功した場合は、以下のような出力が表示されます。

```
FunctionSummary:
  FunctionConfig:
    Comment: Max Age 2 years
    Runtime: cloudfront-js-2.0
  FunctionMetadata:
    CreatedTime: '2021-04-18T21:24:21.314000+00:00'
    FunctionARN: arn:aws:cloudfront::111122223333:function/ExampleFunction
    LastModifiedTime: '2023-12-19T23:41:15.389000+00:00'
    Stage: LIVE
  Name: MaxAge
  Status: UNASSOCIATED
```

関数をディストリビューションに関連付ける

CloudFront Functions の関数を CloudFront ディストリビューションで使用するには、関数をディストリビューションの 1 つ以上のキャッシュ動作に関連付けます。関数を複数のディストリビューションの複数のキャッシュ動作に関連付けることができます。

関数をキャッシュ動作に関連付ける場合は、イベントタイプを選択する必要があります。イベントタイプで、CloudFront Functions がいつ実行されるかが決まります。次のイベントタイプを選択できます。

- ビューワーリクエスト - CloudFront がビューワーからリクエストを受信するとこの関数が実行されます。
- ビューワーレスポンス - CloudFront がビューワーにレスポンスを返す前にこの関数が実行されます。

CloudFront Functions では、オリジン向けのイベントタイプ (オリジンリクエストとオリジンレスポンス) を使用することはできません。代わりに Lambda@Edge を使用できます。詳細については、「Lambda@Edge 関数をトリガーできる CloudFront イベント」を参照してください。

> **Note**
>
> 関数を関連付ける前に、その関数をステージ LIVE に公開する必要があります。

関数をディストリビューションに関連付けるには、CloudFront コンソールまたは AWS Command Line Interface (AWS CLI) を使用できます。

Console

CloudFront コンソールを使用して、既存の CloudFront ディストリビューションの既存のキャッシュ動作に関数を関連付けることができます。ディストリビューションの作成については、「the section called “ディストリビューションを作成する”」を参照してください。

関数を既存のキャッシュ動作に関連付けるには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home#/functions) にサインインし、[関数] ページを選択します。
2. 関連付ける関数を選択します。
3. [関数] ページで、[発行] タブを選択します。
4. [関数を発行] を選択します。
5. [Add association] を選択します。表示されるダイアログで、ディストリビューション、イベントタイプ、および/またはキャッシュ動作を選択します。

   イベントタイプでは、この関数を実行するタイミングを選択します。

   - ビューワーリクエスト – CloudFront がリクエストを受信するたびに関数を実行します。
   - ビューワーレスポンス – CloudFront がレスポンスを返すたびに関数を実行します。
6. 設定を保存するには、[関連付けを追加] を選択します。

CloudFront は関数に ディストリビューションを関連付けます。関連付けられたディストリビューションのデプロイが完了するまで数分待ちます。関数の詳細ページで [ディストリビューションを表示] を選択すると、進行状況を確認できます。

CLI

関数は以下のいずれかに関連付けることができます。

- 既存のキャッシュ動作
- 既存のディストリビューションの新しいキャッシュ動作
- 新しいディストリビューションの新しいキャッシュ動作

次の手順は、関数を既存のキャッシュ動作に関連付ける方法を示しています。

関数を既存のキャッシュ動作に関連付けるには

1. コマンドラインウィンドウを開きます。
2. 次のコードを使用して、関数に関連付けるキャッシュ動作を持つディストリビューションのディストリビューション設定を保存します。このコマンドは、ディストリビューション設定を `dist-config.yaml` という名前のファイルに保存します。このコマンドを使用するには、次の操作を行います。

   - *`DistributionID`* をディストリビューションの ID に置き換えます。
   - コマンドを 1 行で実行します。この例では、例を読みやすくするために改行されています。

   ```
   aws cloudfront get-distribution-config \
       --id DistributionID \
       --output yaml > dist-config.yaml
   ```

   コマンドが正常に完了した場合、AWS CLI は出力を返しません。
3. 作成した `dist-config.yaml` という名前のファイルを開きます。ファイルを編集して以下の変更を加えます。

   a. `ETag` フィールドの名前を `IfMatch` に変更しますが、フィールドの値は変更しないでください。

   b. キャッシュ動作で、`FunctionAssociations` という名前のオブジェクトを見つけます。このオブジェクトを更新して、関数の関連付けを追加します。関数を関連付ける YAML 構文は、次の例のようになります。

- 次の例は、ビューワーリクエストイベントタイプ (トリガー) を示しています。ビューアレスポンスイベントタイプを使用するには、`viewer-request` を `viewer-response` に置き換えます。
- *`arn:aws:cloudfront::111122223333:function/ExampleFunction`* は、このキャッシュ動作に関連付ける関数の Amazon リソースネーム (ARN) に置き換えます。関数 ARN を取得するには、aws cloudfront list-functions コマンドを使用します。

```
FunctionAssociations:
  Items:
    - EventType: viewer-request
      FunctionARN: arn:aws:cloudfront::111122223333:function/ExampleFunction
  Quantity: 1
```

c. 変更が完了したら、ファイルを保存します。

4. 関数の関連付けを追加してディストリビューションを更新するには、次のコマンドを使用します。このコマンドを使用するには、次の操作を行います。

- *`DistributionID`* をディストリビューションの ID に置き換えます。
- コマンドを 1 行で実行します。この例では、例を読みやすくするために改行されています。

```
aws cloudfront update-distribution \
    --id DistributionID \
    --cli-input-yaml file://dist-config.yaml
```

コマンドが正常に完了すると、関数の関連付けで更新されたばかりのディストリビューションを説明する次のような出力が表示されます。次の出力例は、読みやすくするために切り詰めています。

```
Distribution:
  ARN: arn:aws:cloudfront::111122223333:distribution/EBEDLT3BGRBBW
  ... truncated ...
  DistributionConfig:
    ... truncated ...
    DefaultCacheBehavior:
      ... truncated ...
      FunctionAssociations:
```

```
        Items:
        - EventType: viewer-request
          FunctionARN: arn:aws:cloudfront::111122223333:function/ExampleFunction
        Quantity: 1
      ... truncated ...
  DomainName: d111111abcdef8.cloudfront.net
  Id: EDFDVBD6EXAMPLE
  LastModifiedTime: '2021-04-19T22:39:09.158000+00:00'
  Status: InProgress
ETag: E2VJGGQEG1JT8S
```

ディストリビューションが再デプロイされる間に、ディストリビューションの `Status` は `InProgress` に変更されます。新しいディストリビューション設定が CloudFront エッジロケーションに到達するとすぐに、そのエッジロケーションは関連付けられた関数の使用を開始します。ディストリビューションが完全にデプロイされると、`Status` は `Deployed` に戻ります。これは、関連付けられた CloudFront 関数が世界中のすべての CloudFront エッジロケーションでライブされていることを示します。これには通常数分かかります。

Amazon CloudFront KeyValueStore

CloudFront KeyValueStore は、CloudFront Functions 内からの読み取りアクセスを許可する、安全でグローバルな低レイテンシーのキー値データストアであり、CloudFront エッジロケーションで高度かつカスタマイズ可能なロジックを利用できるようにします。

CloudFront KeyValueStore では、関数コードを更新したり、関数に関連付けられているデータを個別に更新できます。このように分離することで関数コードが簡略化され、コードの変更をデプロイしなくてもデータを簡単に更新できます。

> **Note**
>
> CloudFront KeyValueStore を使用するには、CloudFront 関数で JavaScript ランタイム 2.0 を使用する必要があります。

キーと値のペアを使用するための一般的な手順は、次のとおりです。

- キー値ストアを作成して、このストア内に一連のキーと値のペアを設定します。キーバリューストアは、Amazon S3 バケットに追加することも、手動で入力することもできます。
- キー値ストアを CloudFront 関数に関連付けます。

- 関数コード内で、キーの名前を使用してキーに関連付けられた値を取得するか、キーの存在を評価します。キーと値のペアを関数コードで使用する方法と、ヘルパーメソッドの詳細については、「the section called “キー値ストアのヘルパーメソッド”」を参照してください。

CloudFront KeyValueStore の使用を開始する方法の詳細については、AWS ブログ記事「Amazon CloudFront KeyValueStore の紹介」を参照してください。

CloudFront コンソール、CloudFront API、またはサポートされている AWS SDK を使用できます。CloudFront KeyValueStore の使用を開始するには、以下のトピックを参照してください。

トピック

ユースケース

キーと値のペアの一般的なユースケースは以下のとおりです。

- URL の書き換えまたはリダイレクト。キーと値のペアには、書き換えられた URL またはリダイレクト URL を含めることができます。
- A/B テストと機能フラグ。ウェブサイトの特定のバージョンにトラフィックの割合を割り当てることで、テストを実行する関数を作成できます。
- アクセスの許可。独自に定義した条件とキー値ストアに保存したデータに基づいて、リクエストを許可または拒否するアクセスコントロールを実装できます。

サポートされている値の形式

キーと値のペアの値は、以下のいずれかの形式で保存できます。

- 文字列
- バイトでエンコードされた文字列
- JSON

セキュリティ

CloudFront 関数およびすべてのキー値ストアのデータは、次のように安全に処理されます。

- CloudFront KeyValueStore API オペレーションを呼び出すと、CloudFront は保管中および転送中の各キー値ストアを (読み取り時または書き込み時に) 暗号化します。
- 関数を実行すると、CloudFront は CloudFront エッジロケーションでメモリ内のキーと値の各ペアを復号します。

キー値ストアの操作

CloudFront Functions で使用するキーと値のペアを保持するには、キー値ストアを作成する必要があります。

キー値ストアを作成してキーと値のペアを追加すると、CloudFront 関数コードでキー値を使用できます。JavaScript ランタイム 2.0 には、関数コード内のキー値を操作するためのヘルパーメソッドがいくつか含まれています。詳細については、「the section called “キー値ストアのヘルパーメソッド”」を参照してください。

トピック

キー値ストアの作成

空のキー値ストアを作成し、後でキーと値のペアを追加できます。または、キー値ストアおよびキーと値のペアを同時に作成することもできます。

> **Note**
>
> Amazon S3 バケット内のデータソースを指定する場合は、そのバケットに対する `s3:GetObject` と `s3:GetBucketLocation` のアクセス許可が必要です。これらのアクセス許可がない場合、CloudFront はキーバリューストアを正常に作成できません。

Console

キー値ストアを作成するには (コンソールの場合)

1. キー値ストアの作成と同時にキーと値のペアを追加するかどうかを決定します。このインポート機能は、CloudFront コンソールと CloudFront API および AWS SDK の両方でサポートされています。ただし、キー値ストアを最初に作成した場合にのみサポートされます。

   ファイルを使用する場合は、今すぐそれを作成します。
2. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home#/functionsで CloudFront コンソールの [関数] ページを開きます。
3. [KeyValueStores] タブを選択します。[KeyValueStore を作成] を選択します。
4. キー値ストアの名前とオプションの説明を入力します。
5. [S3 URI] を入力します。

   - キーと値のペアのファイルを準備している場合は、ファイルを保存した Amazon S3 バケットへのパスを入力します。
   - キーと値のペアを手動で入力する場合は、このフィールドを空白のままにします。
6. [Create] (作成) を選択します。これで、キーバリューストアが作成されました。

   新しいキー値ストアの詳細ページが表示されます。ページの情報には、キー値ストアの ID と ARN が含まれます。

   - ID は、AWS アカウント内で固有のランダムな文字列です。
   - ARN の構文は次のとおりです。

     ```
     AWS #####:key-value-store/###### ID
     ```
7. [キーと値のペア] セクションを見てください。ファイルをインポートした場合、このセクションにはペアがいくつか表示されます。それ以外の場合は、空です。以下の操作を行うことができます。

- Amazon S3 バケットからファイルをインポートしていない場合、ここでキーと値のペアを追加するには、このセクションに入力できます。
- ファイルをインポートした場合は、値を手動で追加することもできます。
- このセクションを空のままにして、後でキー値ストアを編集してペアを追加することもできます。

ここでペアを追加するには:

- [キーと値のペアを追加] ボタンを選択します。
- [ペアの追加] ボタンを選択して名前と値を入力します。
- [ペアの追加] ボタンを選択してさらにペアを追加します。

完了したら、[変更内容を保存] を選択してキー値ストア内のすべてのペアを保存します。表示される確認ダイアログで [完了] を選択します。

8. ここでキー値ストアを関数に関連付ける場合は、[関連付けられている関数] セクションに入力します。この関連付けは、後で、このキー値ストアの詳細ページや関数の詳細ページから作成することもできます。

   関連付けを今すぐ作成するには、[関数へ移動] ボタンを選択します。詳細については、「???」または「???」を参照してください。

Programmatically

キー値ストアを作成するには

1. キー値ストアの作成と同時にキーと値のペアを追加するかどうかを決定します。キーと値のペアは、後で追加することもできます。このインポート機能は、CloudFront コンソールと CloudFront API および SDK の両方でサポートされています。ただし、キー値ストアを最初に作成した場合にのみサポートされます。

   ファイルを使用する場合は、今すぐそれを作成します。
2. CloudFront API または希望する AWS SDK の create 操作を使用します。例えば、REST API の場合は CloudFront.CreateKeyValueStore を使用します。このオペレーションには複数のパラメータが必要です。
   - 名前。
   - コメントを含む `configuration` パラメータ。

- Amazon S3 バケットに保存したファイルからキーと値のペアをインポートするための `import-source` パラメータ。ファイルからインポートできるのは、キー値ストアを最初に作成している場合のみであることに注意してください。ファイルの形式の詳細については、「the section called “キーと値のペアのファイルの作成”」を参照してください。

オペレーションレスポンスには、以下の情報が含まれます。

- リクエストで渡された値 (割り当てた名前を含む)。
- 作成時間などのデータ。
- ETag (ETVABCEXAMPLE2 など)、キー値ストアの名前を含む ARN (`arn:aws:cloudfront::111122223333:key-value-store/MaxAge` など)。

  キー値ストアをプログラムで操作するには、ETag、ARN、名前をいくつか組み合わせて使用します。

キーバリューストアのステータス

キーバリューストアを作成する場合、データストアのステータス値は次のようになります。

Value	説明
プロビジョニング	キーバリューストアが作成済みであり、指定したデータソースを CloudFront が処理中です。
準備完了	キーバリューストアが作成済みであり、指定したデータソースを CloudFront が正常に処理しました。
インポート失敗	CloudFront は、指定したデータソースを処理できませんでした。このステータスは、ファイルの形式が有効でないか、ファイルのサイズが制限を超えている場合に表示されることがあります。詳細については、「キーと値のペアのファイルの作成」を参照してください。

キー値ストアを関数に関連付ける

キー値ストアを関数に関連付けるには、その関数を操作します。この関連付けは、キー値ストア内のキーと値のペアを関数で使用するために必要です。以下のルールが適用されます。

- 1 つの関数には 1 つのキー値ストアを含めることができます。
- 1 つのキー値ストアを複数の関数に関連付けることができます。

次の方法で関連付けを操作することができます。

- 関数とキー値ストアの間に関連付けを作成することができます。
  - CloudFront コンソールで、キー値ストアの詳細ページを表示し、[関数へ移動] ボタンを選択します。該当するページ、つまり [関数] リスト (現在関連する関数がない場合) または関数の詳細ページ (現在関連付けがある場合) が表示されます。詳細については、「the section called “キー値ストアを関数に関連付ける”」を参照してください。
  - プログラムでは、希望する CloudFront API または SDK の function update オペレーションを使用してください。

  関連付けを作成したら (または関連付けを変更した場合)、関数をテストし、関数を再パブリッシュする必要があります。

- キーと値のペアを変更せずにキー値ストアを変更した場合、関連付けを更新する必要はありません (再発行する必要はありません)。ただし、この関数はテストする必要があります。
- キー値ストア内のキーと値のペアを変更した場合、関連付けを更新する必要はありません (再発行する必要はありません)。ただし、関数をテストして、キーと値のペアの変更に対応していることを確認する必要があります。
- 特定のキー値ストアを使用するすべての関数を表示できます。CloudFront コンソールで、キー値ストアの詳細ページを参照してください。

キー値ストアの変更

キーと値のペアを操作したり、キー値ストアと関数の関連付けを変更したりできます。

Console

キー値ストアを変更するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home#/functions で CloudFront コンソールの [関数] ページを開きます。
2. [KeyValueStores] タブを選択します。変更するキー値ストアを選択します。[詳細] ページが表示されます。

- キーと値のペアを操作するには、[キーと値のペア] セクションで [編集] ボタンを選択します。キーと値のペアを追加または削除したり、既存のキーと値のペアの値を変更したりできます。完了したら、[変更を保存] を選択します。
- このキー値ストアの関連付けを操作するには、[関数へ移動] ボタンを選択します。該当するページ、つまり [関数] リスト (現在関連する関数がない場合) または関数の詳細ページ (現在関連付けがある場合) が表示されます。詳細については、「the section called “キー値ストアを関数に関連付ける”」を参照してください。

Programmatically

キー値ストアは以下の方法で操作できます。

キーと値のペアを変更する

キーと値のペアをさらに追加したり、1 つ以上のキーと値のペアを削除したり、既存のキーと値のペアの値を変更したりできます。詳細については、「the section called “プログラムによるキーと値のペアの操作”」を参照してください。

キー値ストアと関数の関連付けを変更する

このキー値ストアの関連付けを操作するには、「the section called “関数を更新する”」を参照してください。キー値ストアの ARN が必要になります。詳細については、「the section called “キー値ストアへの参照の取得”」を参照してください。

キー値ストアの削除

キー値ストアを削除するには、CloudFront コンソールまたは API を使用できます。

Console

キー値ストアを削除するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home#/functions で CloudFront コンソールの [関数] ページを開きます。
2. キー値ストアが関数に関連付けられているかどうかを確認します。関連付けられている場合は、それを削除します。これらの両方のステップの詳細については、「???」を参照してください。

3. [KeyValueStores] タブを選択します。変更するキーバリューストアを選択し、[削除] を選択します。

Programmatically

キー値ストアを削除するには

1. キー値ストアの ETag と名前を取得します。詳細については、「the section called “キー値ストアへの参照の取得”」を参照してください。
2. キー値ストアが関数に関連付けられているかどうかを確認します。関連付けられている場合は、それを削除します。これらの両方のステップの詳細については、「???」を参照してください。
3. キー値ストアを削除するには、希望する CloudFront API または SDK の delete オペレーションを使用します。例えば、REST API の場合は CloudFront.DeleteKeyValueStore を使用します。

キー値ストアへの参照の取得

キー値ストアをプログラムで操作するには、キー値ストアの ETag と名前が必要です。このデータを取得するには、CloudFront API または希望する AWS SDK を使用して、以下のステップに従います。

1. CloudFront.ListKeyValueStores API 操作を使用して、キー値ストアのリストを返します。変更するキー値ストアの名前を探します。
2. CloudFront.DescribeKeyValueStore API 操作を使用して、前のステップで返したキー値ストアの名前を指定します。

レスポンスには、UUID、キー値ストアの ARN、キー値ストアの ETag が含まれます。

- UUID は 128 ビットです。例えば、`a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`
- ARN には、AWS アカウント番号、定数 `key-value-store`、UUID が含まれます。例:

  ```
  arn:aws:cloudfront::111122223333:key-value-store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
  ```

- ETag は `ETVABCEXAMPLE2` のようになります。

`DescribeKeyValueStore` 操作の詳細については、「the section called “CloudFront KeyValueStore について”」を参照してください。

キーと値のペアのファイルの作成

UTF-8 でエンコードしたファイルを作成するときは、次の JSON 形式を使用します。

```
{
  "data":[
    {
      "key":"key1",
      "value":"value"
    },
    {
      "key":"key2",
      "value":"value"
    }
  ]
}
```

ファイルに重複キーを含めることはできません。Amazon S3 バケットに無効なファイルを指定した場合は、ファイルを更新して重複を削除してから、キーバリューストアを作成し直してみてください。

詳細については、「キー値ストアの作成」を参照してください。

> **Note**
>
> データソースのファイルとそのキーと値のペアには、以下の制限があります。
>
> - ファイルサイズ — 5 MB
> - キーサイズ — 512 文字
> - 値のサイズ — 1024 文字

キーと値のデータでの操作

既存のキー値ストア内のキーと値のペアは、以下の方法で操作できます。

- Amazon CloudFront コンソールの使用。

- CloudFront KeyValueStore API または希望する AWS SDK の使用。

このセクションでは、既存のキー値ストアにキーと値のペアを追加する方法について説明します。キー値ストアを最初に作成するときに、キーと値のペアを含めるには、「the section called “キー値ストアの作成”」を参照してください。

トピック

- CloudFront コンソールを使用したキーと値のペアの操作
- プログラムによるキーと値のペアの操作

CloudFront コンソールを使用したキーと値のペアの操作

CloudFront コンソールを使用してキーと値のペアを操作できます。

キーと値のペアを使用するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home#/functions で CloudFront コンソールの [関数] ページを開きます。
2. [KeyValueStores] タブを選択します。変更するキー値ストアを選択します。[詳細] ページが表示されます。
3. [キーと値のペア] セクションで、[編集] を選択します。
4. キーと値のペアを追加または削除したり、既存のキーと値のペアの値を変更したりできます。
5. 完了したら、[変更を保存] を選択します。

プログラムによるキーと値のペアの操作

> **Note**
>
> CloudFront KeyValueStore API には、CloudFront API とは異なる名前空間があります。

トピック

- キー値ストアへのリファレンスの取得
- キー値ストア内のキーと値のペアの変更
- CloudFront KeyValueStore について

- CloudFront KeyValueStore のコード例

キー値ストアへのリファレンスの取得

CloudFront KeyValueStore を使用して write オペレーションを開始するときは、キー値ストアの ARN と ETag を渡す必要があります。このデータを取得するには、以下を実行します。

1. 希望する CloudFront API または SDK の list オペレーションを使用します。例えば、REST API の場合は CloudFront.ListKeyValueStores を使用します。このレスポンスには、キー値ストアのリストが含まれています。変更するキー値ストアの名前を探します。
2. 希望する CloudFront KeyValueStore API または SDK の describe オペレーションを使用してください。例えば、REST API の場合は CloudFrontKeyValueStore.DescribeKeyValueStore を使用します。前のステップで取得した名前に渡します。

> **Note**
>
> CloudFront API からではなく、CloudFront KeyValueStore API からのオペレーションを使用してください。詳細については、「the section called “CloudFront KeyValueStore について”」を参照してください。

レスポンスには、キー値ストアの ARN と ETag が含まれます。

- ARN には、AWS アカウント番号、定数 `key-value-store`、UUID が含まれます。例:

  ```
  arn:aws:cloudfront::111122223333:key-value-store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
  ```

- ETag は `ETVABCEXAMPLE2` のようになります。

キー値ストア内のキーと値のペアの変更

希望する CloudFront KeyValueStore API または SDK の以下のオペレーションを使用して、キーと値のペアを操作できます。これらのオペレーションはすべて、指定した 1 つのキー値ストアで機能します。

- `CloudFrontKeyValueStore.DeleteKey`: キーを 1 つ削除します。DeleteKey を参照してください。
- `CloudFrontKeyValueStore.GetKey`: キーを 1 つ取得します。GetKey を参照してください。

- `CloudFrontKeyValueStore.ListKeys`: キーを一覧表示します。ListKeys を参照してください。

- `CloudFrontKeyValueStore.PutKey`: 次の 2 つのアクションを実行できます。
  - 1 つのキー値ストアに新しいキーと値のペアを作成する: この場合は、新しいキーの名前と値を渡します。
  - 1 つの既存のキーと値のペアに別の値を設定する: この場合は、既存のキー名と新しいキー値を渡します。

  PutKey を参照してください。
- `CloudFrontKeyValueStore.UpdateKeys`: 1 つの all-or-nothing オペレーションで、次の 1 つ以上のアクションを実行できます。
  - 1 つ以上のキーと値のペアを削除します。
  - 1 つ以上の新しいキーと値のペアを作成します。
  - 1 つまたは複数の既存のキーと値のペアに別の値を設定します。

  UpdateKeys を参照してください。

CloudFront KeyValueStore について

既存のキー値ストア内のキーと値のペアをプログラムで操作するには、CloudFront KeyValueStore サービスを使用します。

キー値ストアを最初に作成するときに、いくつかのキーと値のペアを含めるには、CloudFront サービスを使用します。

describe オペレーション

CloudFront API と CloudFront KeyValueStore API の両方に、キー値ストアに関するデータを返す describe オペレーションがあります。

- CloudFront API は、ストア自体が最後に変更されたステータスや日付などのデータを提供します。
- CloudFront KeyValueStore API は、ストレージリソースのコンテンツに関するデータ (ストア内のキーと値のペアやコンテンツのサイズ) を提供します。

この 2 つの API の describe オペレーションは、キー値ストアを識別する若干異なるデータを返します。

- CloudFront API の describe オペレーションは、キー値ストアの ETag、UUID、ARN を返します。
- CloudFront KeyValueStore API の describe オペレーションは、キー値ストアの ETag と ARN を返します。

> **Note**
>
> describe オペレーションはそれぞれ異なる ETag を返します。ETag は置き換え可能ではありません。
> いずれかの API でオペレーションを実行する場合、適切な API から ETag を渡す必要があります。例えば、CloudFront KeyValueStore の delete オペレーションでは、CloudFront KeyValueStore の describe オペレーションから取得した ETag を渡します。

CloudFront KeyValueStore のコード例

Example : **`DescribeKeyValueStore`** API オペレーションを呼び出す

次のコード例は、キーバリューストアの `DescribeKeyValueStore` API オペレーションを呼び出す方法を示しています。

```
const {
  CloudFrontKeyValueStoreClient,
  DescribeKeyValueStoreCommand,
} = require("@aws-sdk/client-cloudfront-keyvaluestore");

require("@aws-sdk/signature-v4-crt");

(async () => {
  try {
    const client = new CloudFrontKeyValueStoreClient({
      region: "us-east-1"
    });
    const input = {
      KvsARN: "arn:aws:cloudfront::123456789012:key-value-store/a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111",
    };
```

```
    const command = new DescribeKeyValueStoreCommand(input);

    const response = await client.send(command);
  } catch (e) {
    console.log(e);
  }
})();
```

Lambda@Edge を使用してエッジでカスタマイズする

Lambda@Edge は AWS Lambda の拡張です。Lambda@Edge は、Amazon CloudFront が配信するコンテンツをカスタマイズする関数を実行できるコンピューティングサービスです。Node.js または Python 関数は、Lambda コンソールで 1 つの AWS リージョン、米国東部 (バージニア北部) で作成できます。

次に、サーバーをプロビジョニングまたは管理することなく、ビューワーに近い AWS の場所で関数を実行する Lambda または CloudFront コンソールにトリガーを追加します。オプションで、Lambda および CloudFront API オペレーションを使用して、関数とトリガーをプログラムで設定できます。

Lambda@Edge は、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケーリングします。オリジンサーバーの代わりに、ビューワーの最寄りの AWS ロケーションでリクエストを処理すると、レイテンシーが大幅に軽減され、ユーザーエクスペリエンスが向上します。

トピック

- Lambda@Edge がリクエストとレスポンスでどのように機能するかを説明します。
- Lambda@Edge の使用方法
- Lambda@Edge 関数の使用を開始する
- Lambda@Edge 用の IAM アクセス許可とロールのセットアップ
- Lambda@Edge 関数を記述および作成する
- Lambda@Edge 関数のトリガーを追加する
- Lambda@Edge 関数をテストおよびデバッグする
- Lambda@Edge 関数とレプリカを削除する
- Lambda@Edge イベント構造
- リクエストとレスポンスを使用する
- Lambda@Edge 関数の例

Lambda@Edge がリクエストとレスポンスでどのように機能するかを説明します。

CloudFront ディストリビューションを Lambda@Edge 関数に関連付けると、CloudFront エッジロケーションでリクエストとレスポンスがインターセプトされます。Lambda 関数は、次の CloudFront イベントの発生時に実行できます。

- CloudFront がビューワーからリクエストを受信したとき (ビューワーリクエスト)
- CloudFront がリクエストをオリジンに転送する前 (オリジンリクエスト)
- CloudFront がオリジンからレスポンスを受信したとき (オリジンレスポンス)
- CloudFront がビューワーにレスポンスを返す前 (ビューワーレスポンス)

AWS WAF を使用している場合、Lambda@Edge ビューワーリクエストは AWS WAF ルールの適用後に実行されます。

詳細については、リクエストとレスポンスを使用するおよびLambda@Edge イベント構造を参照してください。

Lambda@Edge の使用方法

Amazon CloudFront ディストリビューションでの Lambda@Edge 処理には多くの用途があります。例:

- Lambda 関数 は Cookie を検査して URL を書き換え、A/B テスト用に異なるバージョンのサイトをユーザーに表示できます。
- CloudFront は、デバイスに関する情報が含まれている `User-Agent` ヘッダーを確認し、ビューワーが使用しているデバイスに基づいて、異なるオブジェクトをビューワーに返すことができます。例えば、CloudFront は、デバイスの画面サイズに基づいて異なるイメージを返すことができます。同様に、`Referer` ヘッダーの値を考慮する関数を作成して、CloudFront が最も低い解像度のイメージをボットに返すこともできます。
- または、他の条件の Cookie を確認できます。例えば、衣料品の通販ウェブサイトで、ユーザーが選択したジャケットの色を Cookie に保存する場合に、Lambda 関数でリクエストを変更して、CloudFront が選択した色のジャケットのイメージを返すようにできます。
- Lambda 関数は、CloudFront のビューワーリクエストイベントまたはオリジンリクエストイベントの発生時に HTTP レスポンスを生成できます。

- 関数でヘッダーまたは認証トークンを検査して、CloudFront がリクエストをオリジンに転送する前に、コンテンツへのアクセスを制御するヘッダーを挿入できます。
- Lambda 関数は、外部リソースのネットワーク呼び出しを実行して、ユーザー認証情報を確認したり、追加のコンテンツを取得してレスポンスをカスタマイズしたりすることもできます。

サンプルコードを含むその他のヒントについては、「Lambda@Edge 関数の例」を参照してください。

コンソールで Lambda@Edge を設定する手順については、「チュートリアル: 基本的な Lambda@Edge 関数を作成する」を参照してください。

Lambda@Edge 関数の使用を開始する

Lambda@Edge では、CloudFront トリガーを使用して Lambda 関数を呼び出すことができます。CloudFront ディストリビューションを Lambda 関数に関連付けると、CloudFront エッジロケーションでリクエストとレスポンスがインターセプトされ、関数が実行されます。Lambda 関数では、セキュリティを強化したり、ビューワーに近い情報をカスタマイズしてパフォーマンスを向上させたりできます。

次のリストは、CloudFront で Lambda 関数を作成および使用する方法の基本的な概要を示しています。ステップバイステップのチュートリアルについては、「チュートリアル: 基本的な Lambda@Edge 関数を作成する」を参照してください。

1. AWS Lambda コンソールで、米国東部 (バージニア北部) リージョンに Lambda 関数を作成します (いずれかの AWS SDK を使用してプログラムで関数を作成することもできます)。
2. 番号付きバージョンの関数を保存して発行します。

   関数を変更する場合、米国東部 (バージニア北部) リージョンで関数の $LATEST バージョンを編集する必要があります。次に、CloudFront と連携するように設定する前に、新しい番号付きバージョンを発行します。
3. 関数を CloudFront ディストリビューションとキャッシュビヘイビアに関連付けます。1 つまたは複数の CloudFront イベント (トリガー) を指定します。これにより、関数を実行できます。例えば、CloudFront がビューワーからリクエストを受け取ったときに関数を実行させるトリガーを作成できます。
4. トリガーを作成すると、Lambda が世界中の AWS ロケーションに関数のレプリカを作成します。

> **Tip**
>
> 独自のカスタムソリューションで Lambda@Edge をどのように使用できるかについて詳しく説明します。関数の作成と更新、イベントの構造、および CloudFront トリガーの追加に関する解説をご覧ください。また、さらに多くのアイデアやコードサンプルを「Lambda@Edge 関数の例」で参照できます。

トピック

チュートリアル: 基本的な Lambda@Edge 関数を作成する

このチュートリアルでは、CloudFront で実行されるサンプル Node.js 関数を作成および設定することで、Lambda@Edge の使用を開始する方法を示します。この例では、CloudFront がファイルを取得するときに、セキュリティヘッダーがレスポンスに追加されます。(これにより、ウェブサイトのセキュリティとプライバシーが向上します)。

このチュートリアルでは、独自のウェブサイトは必要ありません。しかしながら、独自の Lambda@Edge ソリューションを作成するときは、同様のステップに従って同じオプションから選択できます。

トピック

ステップ 1: AWS アカウント にサインアップする

まだサインアップしていない場合は、AWS アカウントにサインアップします。詳細については、「AWS アカウントへのサインアップ」を参照してください。

ステップ 2: CloudFront ディストリビューションを作成する

Lambda@Edge 関数の例を作成する前に、コンテンツの提供元のオリジンを含む、操作対象の CloudFront 環境が必要です。

この例では、Amazon S3 バケットをディストリビューションのオリジンとして使用する CloudFront ディストリビューションを作成します。使用する環境が既にある場合、このステップは省略できます。

Amazon S3 オリジンを使用して CloudFront ディストリビューションを作成するには

1. サンプルコンテンツ用に、イメージファイルなど 1～2 つのファイルで Amazon S3 バケットを作成します。そのためには、コンテンツを Amazon S3 にアップロードするステップに従います。必ず、バケットのオブジェクトへのパブリック読み取りアクセス権を付与するアクセス許可を設定します。
2. CloudFront ウェブディストリビューションを作成するステップに従って、CloudFront ディストリビューションを作成し、オリジンとして S3 バケットを追加します。ディストリビューションが既にある場合は、代わりにそのディストリビューションのオリジンとしてバケットを追加できます。

> **Tip**
>
> ディストリビューション ID をメモします。このチュートリアルで後ほど関数の CloudFront トリガーを追加する場合は、ドロップダウンリストでディストリビューションの ID (E653W22221KDDL など) を選択する必要があります。

ステップ 3: 関数を作成する

このステップでは、Lambda コンソールでブループリントテンプレートから Lambda 関数を作成します。関数コードは、CloudFront ディストリビューションでセキュリティヘッダーを更新するコードを追加します。

Lambda 関数を作成するには

1. AWS Management Console にサインインして、AWS Lambda で https://console.aws.amazon.com/lambda/ コンソールを開きます。

> **Important**
>
> US-East-1 (バージニア北部) AWS リージョン (us-east-1) にいることを確認します。Lambda@Edge 関数を作成するには、このリージョンに設定されている必要があります。

2. [関数を作成] を選択します。
3. [関数の作成] ページで、[設計図の使用] を選択し、各フィールドに「**cloudfront**」と入力して CloudFront の設計図をフィルタリングします。

> **Note**
>
> CloudFront の設計図は、US-East-1 (バージニア北部) リージョン (us-east-1) でのみ使用できます。

4. 関数のテンプレートとして [HTTP レスポンスヘッダーを変更する] ブループリントを選択します。
5. 関数に関する次の情報を入力します。

関数名

関数の名前を入力します。

実行ロール

関数のアクセス許可を設定する方法を選択します。推奨される基本 Lambda@Edge アクセス許可ポリシーテンプレートを使用するには、[AWS ポリシーテンプレートから新しいロールを作成する] を選択します。

ロール名

ポリシーテンプレートが作成するロールの名前を入力します。

ポリシーテンプレート

関数の基盤として CloudFront ブループリントを選択したことにより、Lambda は、ポリシーテンプレート [基本的な Lambda@Edge アクセス許可] を自動的に追加します。このポリシーテンプレートでは、世界中の CloudFront の場所で、Lambda 関数の実行を CloudFront に許可する実行ロールアクセス許可を追加します。詳細については、「Lambda@Edge 用の IAM アクセス許可とロールのセットアップ」を参照してください。

6. [Create function (関数の作成)] を選択します。
7. 表示される [Lambda@Edge へのデプロイ] ペインで、[キャンセル] を選択します (このチュートリアルでは、関数を Lambda@Edge にデプロイする前に関数コードを変更する必要があります)。
8. ページの [コードソース] セクションまで下にスクロールします。
9. テンプレートコードを、オリジンが返すセキュリティヘッダーを変更する関数に置き換えます。たとえば、以下のようなコードを実行できます。

```
'use strict';
exports.handler = (event, context, callback) => {

    //Get contents of response
    const response = event.Records[0].cf.request;
    const headers = response.headers;

    //Set new headers
    headers['strict-transport-security'] = [{key: 'Strict-Transport-Security',
 value: 'max-age= 63072000; includeSubdomains; preload'}];
    headers['content-security-policy'] = [{key: 'Content-Security-Policy', value:
 "default-src 'none'; img-src 'self'; script-src 'self'; style-src 'self'; object-
src 'none'"}];
    headers['x-content-type-options'] = [{key: 'X-Content-Type-Options', value:
 'nosniff'}];
    headers['x-frame-options'] = [{key: 'X-Frame-Options', value: 'DENY'}];
    headers['x-xss-protection'] = [{key: 'X-XSS-Protection', value: '1;
 mode=block'}];
    headers['referrer-policy'] = [{key: 'Referrer-Policy', value: 'same-origin'}];

    //Return modified response
    callback(null, response);
};
```

10. [ファイル]、[保存] の順に選択して、更新したコードを保存します。

次のセクションに進み、関数を実行する CloudFront トリガーを追加します。

ステップ 4: 関数を実行する CloudFront トリガーを追加する

これでセキュリティヘッダーを更新する Lambda 関数ができたので、CloudFront がディストリビューションのオリジンから受け取ったヘッダーをレスポンスに追加するように、関数を実行する CloudFront トリガーを設定します。

関数の CloudFront トリガーを設定するには

1. Lambda コンソールで、関数の [関数の概要] ページの [トリガーを追加] を選択します。
2. [トリガー設定] で、[CloudFront] を選択します。
3. [Lambda@Edge へのデプロイ] を選択します。
4. [Lambda@Edge へのデプロイ] ペインの [CloudFront トリガーの設定] で、次の情報を入力します。

   ディストリビューション

   関数に関連付ける CloudFront ディストリビューション ID。ドロップダウンリストで、ディストリビューション ID を選択します。

   キャッシュ動作

   トリガーで使用するキャッシュ動作。この例では、値を * に設定したままにします。これは、ディストリビューションのデフォルトのキャッシュ動作を意味します。詳細については、「ディストリビューション設定リファレンス」トピックの「キャッシュ動作の設定」を参照してください。

   CloudFront イベント

   関数をいつ実行するか指定するトリガー。CloudFront がオリジンからレスポンスを返すたびに、セキュリティヘッダー関数を実行したいと思います。したがって、ドロップダウンリストで [オリジンレスポンス] を選択します。詳細については、「Lambda@Edge 関数のトリガーを追加する」を参照してください。
5. [Lambda@Edge へのデプロイを確認] チェックボックスをオンにします。
6. [デプロイ] を選択して、トリガーを追加し、世界中の AWS の場所に関数をレプリケートします。
7. 関数がレプリケートするまで待ちます。これには通常数分かかります。

   レプリケーションが終了したかどうかを確認するには、CloudFront コンソールに移動し、ディストリビューションを表示します。ディストリビューションのステータスが [デプロイ中] から日時に変わるまで待ち、関数がレプリケートされたことを確認します。関数が機能することを確認するには、次のセクションのステップに従います。

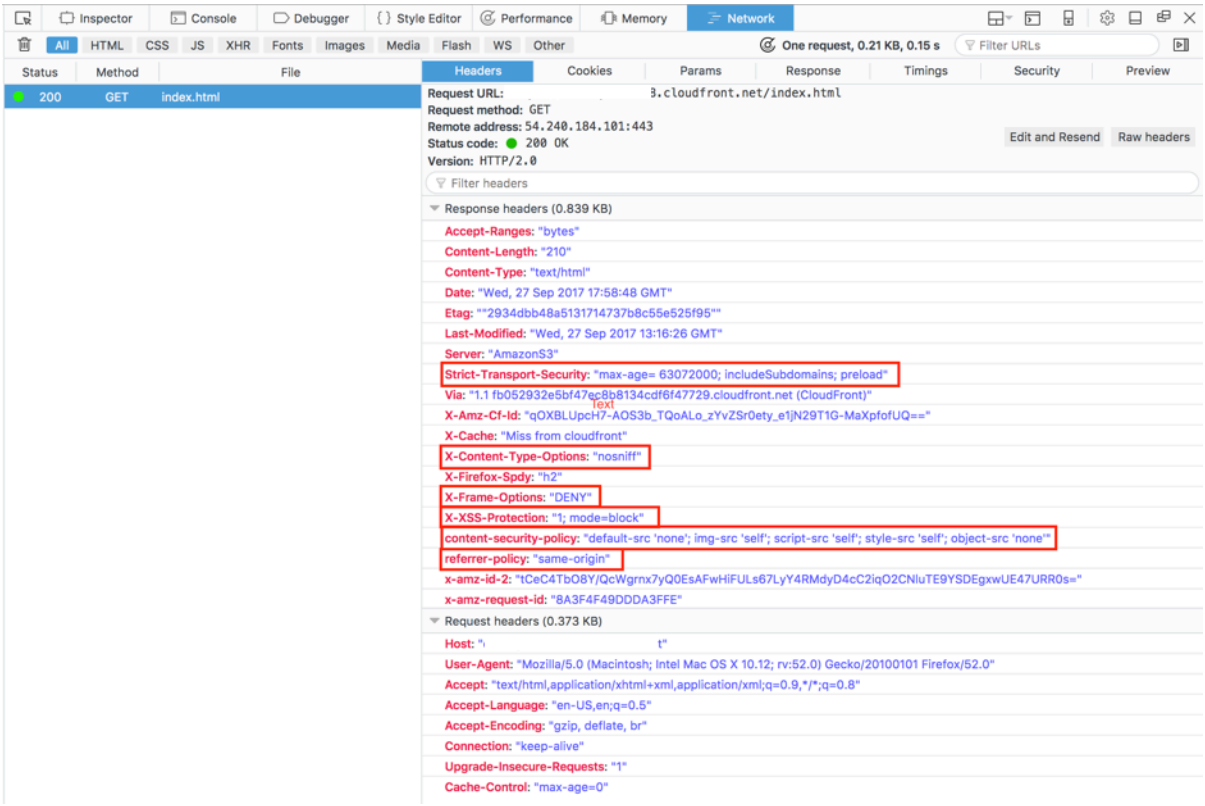
ステップ 5: 関数の実行を確認する

Lambda 関数を作成し、CloudFront ディストリビューションに対してその関数を実行するトリガーが設定されたため、関数が正常に動作することを確認します。この例では、CloudFront が返す HTTP ヘッダーで、セキュリティヘッダーが追加されることを確認します。

Lambda@Edge 関数でセキュリティヘッダーが追加されることを確認するには

1. ブラウザで、S3 バケット内のファイルの URL を入力します。たとえば、`https://d111111abcdef8.cloudfront.net/image.jpg` のような URL を使用できます。

   ファイル URL で使用する CloudFront ドメイン名の詳細については、「CloudFront でファイルの URL 形式をカスタマイズする」を参照してください。
2. ブラウザのウェブデベロッパーツールバーを開きます。たとえば、Chrome のブラウザウィンドウで、コンテキスト (右クリック) メニューを開き、[Inspect (調査)] を選択します。
3. [Network (ネットワーク)] タブを選択します。
4. ページを再ロードしてイメージを表示し、左側のペインの HTTP リクエストを選択します。HTTP ヘッダーが別のペインに表示されます。
5. HTTP ヘッダーのリストを確認し、予期されるセキュリティヘッダーがリストに含まれていることを確認します。たとえば、次のスクリーンショットに示すようなヘッダーが表示されます。



セキュリティヘッダーがヘッダーのリストに含まれていれば、成功です。最初の Lambda@Edge 関数を正常に作成しました。CloudFront が エラーを返す場合や、その他の問題がある場合は、次のステップに進んで問題のトラブルシューティングを行います。

ステップ 6: 問題のトラブルシューティングを行う

CloudFront が エラーを返すか、予期どおりにセキュリティヘッダーを追加しない場合は、CloudWatch Logs を参照して関数の実行を調査できます。必ず、関数が実行された場所に最も近い AWS ロケーションで保存されたログを使用します。

例えば、ロンドンからファイルを表示する場合は、CloudWatch コンソールでリージョンを欧州 (ロンドン) に変更してみてください。

Lambda@Edge 関数の CloudWatch Logs を調べるには

1. AWS Management Console にサインインして、CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. [リージョン] を、ブラウザでファイルを表示したときに表示されるロケーションに変更します。これは、関数が実行されている場所です。
3. 左側のペインで、[ログ] を選択して、ディストリビューションのログを表示します。

詳細については、「Amazon CloudWatch による CloudFront メトリクスのモニタリング」を参照してください。

ステップ 7: リソース例をクリーンアップする

このチュートリアルのためだけに Amazon S3 バケットと CloudFront ディストリビューションを作成した場合は、割り当てた AWS リソースを削除して、今後料金が発生しないようにしてください。AWS リソースを削除すると、追加したコンテンツは使用できなくなります。

タスク

- S3 バケットの削除
- Lambda 関数を削除する
- CloudFront ディストリビューションの削除

S3 バケットの削除

Amazon S3 バケットを削除する前に、バケットのログ記録が無効であることを確認します。それ以外の場合、削除するバケットへのログの書き込みが AWS によって継続されます。

バケットのログ記録を無効にするには

1. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. バケットを選択し、[プロパティ] を選択します。
3. [プロパティ] から [ログ記録] を選択します。
4. [有効] チェックボックスをオフにします。
5. [Save] を選択します。

これで、バケットを削除できます。詳細については、Amazon Simple Storage Service コンソールユーザーガイドの「バケットの削除」を参照してください。

Lambda 関数を削除する

Lambda 関数の関連付けを削除する手順と、関数自体を削除する手順 (オプション) については、「Lambda@Edge 関数とレプリカを削除する」を参照してください。

CloudFront ディストリビューションの削除

CloudFront ディストリビューションを削除する前に、ディストリビューションを無効にする必要があります。無効になったディストリビューションは機能しなくなり、料金も発生しません。無効にしたディストリビューションはいつでも有効にすることができます。無効にしたディストリビューションを削除すると、使用できなくなります。

CloudFront ディストリビューションを無効にして削除するには

1. で CloudFront コンソールを開きますhttps://console.aws.amazon.com/cloudfront/v4/home
2. 無効にするディストリビューションを選択してから [Disable (無効化)] を選択します。
3. 確認を求められたら、[Yes, Disable (はい、無効化する)] を選択します。
4. 無効にしたディストリビューションを選択してから [削除] を選択します。
5. 確認を求めるメッセージが表示されたら、[Yes, Delete (はい、削除します)] を選択します。

その他のリソース

Lambda@Edge 関数の動作について基本的な理解を得たので、以下を参照してさらに詳しく学習します。

- Lambda@Edge 関数の例
- Lambda@Edge 設計のベストプラクティス
- Lambda@Edge を使用したレイテンシーの軽減とエッジへのコンピューティングの移行

Lambda@Edge 用の IAM アクセス許可とロールのセットアップ

Lambda@Edge を設定するには、Lambda に対する次の IAM アクセス許可とロールが必要です。

- IAM アクセス許可 – これらのアクセス許可により、AWS Lambda 関数を作成し、CloudFront ディストリビューションに関連付けることができます。
- Lambda 関数実行ロール (IAM ロール) – Lambda サービスプリンシパルは、このロールを引き受けて関数を実行します。
- Lambda@Edge のサービスリンクロール – サービスリンクロールにより、特定の AWS のサービス が Lambda 関数を AWS リージョン にレプリケートし、CloudWatch が CloudFront ログファイルを使用できるようになります。

Lambda@Edge 関数を CloudFront ディストリビューションに関連付けるために必要な IAM アクセス許可

Lambda に必要な IAM アクセス許可に加え、ユーザーは、Lambda 関数を CloudFront ディストリビューションに関連付けるための以下の IAM アクセス許可が必要です。

- `lambda:GetFunction` – Lambda 関数の設定情報を取得するためのアクセス許可、およびその関数を含む `.zip` ファイルをダウンロードするための署名付き URL を取得するアクセス許可を付与します。
- `lambda:EnableReplication*` – Lambda レプリケーションサービスが関数コードと設定を取得するためのアクセス許可をリソースポリシーに付与します。
- `lambda:DisableReplication*` – Lambda レプリケーションサービスが関数を削除するためのアクセス許可をリソースポリシーに付与します。

> **Important**
>
> `lambda:EnableReplication*` および `lambda:DisableReplication*` アクションの最後にアスタリスク (*) を追加する必要があります。

- リソースに対して、次の例のように、CloudFront イベントが発生した場合に実行する関数バージョンの ARN を指定します。

  `arn:aws:lambda:us-east-1:123456789012:function:TestFunction:2`

- `iam:CreateServiceLinkedRole` – Lambda@Edge が CloudFront で Lambda 関数をレプリケートするために使用するサービスリンクロールを作成するアクセス許可を付与します。Lambda@Edge を初めて設定すると、サービスリンクロールが自動的に作成されます。Lambda@Edge を使用する他のディストリビューションにこのアクセス許可を追加する必要はありません。

- `cloudfront:UpdateDistribution` または `cloudfront:CreateDistribution` – ディストリビューションを更新または作成するアクセス許可を付与します。

詳細については、次のトピックを参照してください。

- Amazon CloudFront のアイデンティティとアクセス管理
- 「AWS Lambda デベロッパーガイド」の「Lambda リソースのアクセス許可」

サービスプリンシパルの関数実行ロール

ユーザーの関数を実行するときに `lambda.amazonaws.com` と `edgelambda.amazonaws.com` サービスプリンシパル が引き受けることができる IAM ロールを作成する必要があります。

> **Tip**
>
> Lambda コンソールで関数を作成する場合、AWS ポリシーテンプレートを使用して新しい実行ロールを作成することを選択できます。このステップでは、関数を実行するために必要な Lambda@Edge アクセス許可が自動的に追加されます。チュートリアル: シンプルな Lambda@Edge 関数の作成のステップ 5 を参照してください。

IAM ロールを手動で作成する詳細については、「IAM ユーザーガイド」の「ロールの作成とポリシーのアタッチ (コンソール)」を参照してください。

Example 例: ロール信頼ポリシー

IAM コンソールの [信頼関係] タブで、このロールを追加できます。このポリシーは [アクセス許可] タブには追加しないでください。

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

実行ロールに付与する必要がある許可の詳細については、「AWS Lambda デベロッパーガイド」の「Lambda リソースのアクセス許可」を参照してください。

メモ

- デフォルトでは、CloudFront イベントが Lambda 関数をトリガーするたびに、データが CloudWatch Logs に書き込まれます。これらのログを使用する場合は、CloudWatch Logs にデータを書き込むためのアクセス権限が実行ロールに必要です。事前定義された AWSLambdaBasicExecutionRole を使用して、実行ロールにアクセス許可を付与できます。

  CloudWatch Logs の詳細については、「the section called “エッジ関数のログ”」を参照してください。

- S3 バケットからのオブジェクトの読み取りなど、Lambda 関数コードが他の AWS リソースにアクセスする場合、そのアクションを実行するためのアクセス許可が実行ロールに必要です。

Lambda@Edge 用のサービスにリンクされたロール

Lambda@Edge は IAM サービスリンクロールを使用します。サービスにリンクされたロールは、サービスに直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、サービスによって事前定義されており、お客様の代わりにサービスから他の AWS サービスを呼び出す必要のあるアクセス許可がすべて含まれています。

Lambda@Edge は、以下の IAM サービスリンクロールを使用します。

- AWSServiceRoleForLambdaReplicator - Lambda@Edge はこのロールを使用して、Lambda@Edge が関数を AWS リージョン にレプリケートできるようにします。

  CloudFront で Lambda@Edge トリガーを初めて追加すると、AWSServiceRoleForLambdaReplicator という名前のロールが自動的に作成され、Lambda@Edge が関数を AWS リージョン にレプリケートできるようになります。このロールは、Lambda@Edge 関数を使用するために必要です。AWSServiceRoleForLambdaReplicator ロールの ARN は次の例のようになります。

  ```
  arn:aws:iam::123456789012:role/aws-service-role/
  replicator.lambda.amazonaws.com/AWSServiceRoleForLambdaReplicator
  ```

- AWSServiceRoleForCloudFrontLogger – CloudFront はこのロールを使用してログファイルを CloudWatch にプッシュします。ログファイルを使用して Lambda@Edge 検証エラーをデバッグできます。

  Lambda@Edge 関数の関連付けを追加すると、AWSServiceRoleForCloudFrontLogger ロールが自動的に作成され、CloudFront が Lambda@Edge エラーログファイルを CloudWatch にプッシュできるようになります。AWSServiceRoleForCloudFrontLogger の ARN は次のようになります。

  ```
  arn:aws:iam::account_number:role/aws-service-role/
  logger.cloudfront.amazonaws.com/AWSServiceRoleForCloudFrontLogger
  ```

サービスリンクロールを使用することで、必要なアクセス許可を手動で追加する必要がなくなるため、Lambda@Edge のセットアップと使用が簡単になります。Lambda@Edge はそのサービスリ

ンクロールのアクセス許可を定義し、Lambda@Edge のみがそのロールを引き受けることができます。定義されたアクセス権限には、信頼ポリシーとアクセス権限ポリシーが含まれます。その他の IAM エンティティにアクセス許可ポリシーをアタッチすることはできません。

サービスにリンクされたロールを削除するには、その前に、それらのロールに関連付けられている CloudFront または Lambda@Edge のリソースを削除する必要があります。このようにして、アクティブなリソースにアクセスするためにまだ必要な、サービスリンクロールを削除しないようにすることで、Lambda@Edge リソースが保護されます。

サービスにリンクされたロールの詳細については、「CloudFront のサービスにリンクされたロール」を参照してください。

Lambda@Edge 用のサービスにリンクされたロールのアクセス許可

Lambda@Edge は、AWSServiceRoleForLambdaReplicator および AWSServiceRoleForCloudFrontLogger という名前の 2 つのサービスにリンクされたロールを使用します。以下のセクションでは、それらの各ロールのアクセス許可を管理する方法について説明します。

目次

Lambda Replicator 用のサービスにリンクされたロールのアクセス許可

このサービスにリンクされたロールにより、Lambda が Lambda@Edge 関数を AWS リージョン にレプリケートできるようになります。

AWSServiceRoleForLambdaReplicator サービスにリンクされたロールは、ロールを継承するために `replicator.lambda.amazonaws.com` のサービスを信頼します。

このロールのアクセス権限ポリシーは、Lambda@Edge が以下のアクションを指定されたリソースに対して実行することを許可します。

- `lambda:CreateFunction` `arn:aws:lambda:*:*:function:*` での
- `lambda:DeleteFunction` `arn:aws:lambda:*:*:function:*` での
- `lambda:DisableReplication` `arn:aws:lambda:*:*:function:*` での
- `iam:PassRole` `all AWS resources` での
- `cloudfront:ListDistributionsByLambdaFunction` `all AWS resources` での

CloudFront ロガー用のサービスにリンクされたロールのアクセス許可

このサービスリンクロールでは、Lambda@Edge の検証エラーをデバッグするのに役立つように CloudFront が CloudWatch にログファイルをプッシュすることが許可されます。

AWSServiceRoleForCloudFrontLogger サービスにリンクされたロールは、ロールを継承するために `logger.cloudfront.amazonaws.com` のサービスを信頼します。

このロールのアクセス権限ポリシーは、Lambda@Edge が以下のアクションを指定された `arn:aws:logs:*:*:log-group:/aws/cloudfront/*` リソースに対して実行することを許可します。

- `logs:CreateLogGroup`
- `logs:CreateLogStream`
- `logs:PutLogEvents`

IAM エンティティ (ユーザー、グループ、ロールなど) で Lambda@Edge のサービスにリンクされたロールを削除できるように、アクセス許可を設定する必要があります。詳細については、「IAM User Guide」(IAM ユーザーガイド) の「Service-linked role permissions」(サービスにリンクされたロールのアクセス権限) を参照してください。

Lambda@Edge 用のサービスにリンクされたロールの作成

通常、Lambda@Edge のサービスにリンクされたロールを手動で作成することはありません。以下のシナリオで、サービスによってロールが自動的に作成されます。

- トリガーを初めて作成するとき、サービスは AWSServiceRoleForLambdaReplicator ロールを作成します (まだ存在しない場合) 。このロールにより、Lambda が Lambda@Edge 関数を AWS リージョン にレプリケートできるようになります。

  このサービスにリンクされたロールを削除した場合、Lambda@Edge の新しいトリガーをディストリビューションに追加すると、そのロールは再び作成されます。
- Lambda@Edge が関連付けられた CloudFront ディストリビューションを更新または作成すると、サービスによって AWSServiceRoleForCloudFrontLogger ロールが作成されます (まだ存在しない場合)。このロールにより、CloudFront が CloudWatch にログファイルをプッシュできるようになります。

  このサービスリンクロールを削除した場合は、Lambda@Edge の関連付けがある CloudFront ディストリビューションを更新または作成すると、そのロールが再び作成されます。

これらのサービスリンクロールを手動で作成する必要がある場合は、次の AWS Command Line Interface (AWS CLI) コマンドを実行します。

AWSServiceRoleForLambdaReplicator ロールを作成するには

- 以下のコマンドを実行します。

```
aws iam create-service-linked-role --aws-service-name
 replicator.lambda.amazonaws.com
```

AWSServiceRoleForCloudFrontLogger ロールを作成するには

- 以下のコマンドを実行します。

```
aws iam create-service-linked-role --aws-service-name
 logger.cloudfront.amazonaws.com
```

Lambda@Edge のサービスにリンクされたロールの編集

Lambda@Edge のサービスリンクロール AWSServiceRoleForLambdaReplicator または AWSServiceRoleForCloudFrontLogger を編集することはできません。サービスによってサービスリンクロールが作成された後は、多くのエンティティでそのロールが参照されるため、そのロール名は変更できません。ただし、IAM を使用してロールの説明を編集することはできます。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの編集」を参照してください。

CloudFront のサービスにリンクされたロールでサポートされる AWS リージョン

CloudFront は、次の AWS リージョン で Lambda@Edge 用のサービスにリンクされたロールの使用をサポートしています。

- 米国東部 (バージニア北部) – us-east-1
- 米国東部 (オハイオ) – us-east-2
- 米国西部 (北カリフォルニア) – us-west-1
- 米国西部 (オレゴン) – us-west-2
- アジアパシフィック (ムンバイ) – ap-south-1
- アジアパシフィック (ソウル) – ap-northeast-2
- アジアパシフィック (シンガポール) – ap-southeast-1

- アジアパシフィック (シドニー) – ap-southeast-2
- アジアパシフィック (東京) – ap-northeast-1
- 欧州 (フランクフルト) – eu-central-1
- 欧州 (アイルランド) – eu-west-1
- 欧州 (ロンドン) – eu-west-2
- 南米 (サンパウロ) – sa-east-1

Lambda@Edge 関数を記述および作成する

Lambda@Edge を使用するには、AWS Lambda 関数のコードを記述します。次に、トリガーと呼ばれる特定の CloudFront イベントに基づいて関数を実行するように Lambda を設定します。

AWS Management Console を使用して、Lambda 関数と CloudFront トリガーを操作できます。または、API を使用してプログラムで Lambda@Edge を操作できます。

トピック

Lambda@Edge 関数を記述する

Lambda@Edge 関数の記述については、以下のリソースを参照してください。

- Lambda@Edge イベント構造 – Lambda@Edge で使用するイベント構造を理解します。
- Lambda@Edge 関数の例 – A/B テストや HTTP リダイレクトの生成などの関数の例。

Lambda@Edge で Node.js または Python を使用するためのプログラミングモデルは、AWS リージョン で Lambda を使用するプログラミングモデルと同じです。詳細については、「AWS Lambda デベロッパーガイド」の「Node.js を使用した Lambda 関数の作成」または「Python を使用した Lambda 関数の作成」を参照してください。

Lambda@Edge 関数で、`callback` パラメータを含めて、リクエストまたはレスポンスイベントの該当するオブジェクトを返します。

- リクエストイベント - レスポンスに `cf.request` オブジェクトを含めます。

レスポンスを生成している場合は、レスポンスに cf.response オブジェクトを含めます。詳細については、「リクエストトリガーでの HTTP レスポンスを生成する」を参照してください。

- レスポンスイベント - レスポンスに cf.response オブジェクトを含めます。

Lambda@Edge 関数を作成する

CloudFront イベントに基づく Lambda 関数を実行するように AWS Lambda をセットアップするには、以下の手順に従います。

Lambda@Edge 関数を作成するには (コンソール)

1. AWS Management Console にサインインして AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. 1 つ以上の Lambda 関数が既にある場合は、[Create function] を選択します。

   関数がない場合は、[Get Started Now] を選択します。
3. ページの上部にあるリージョンのリストで、[米国東部 (バージニア北部)] を選択します。
4. 独自のコードを使用して関数を作成するか、CloudFront の設計図で始まる関数を作成します。

   - 独自のコードを使用して関数を作成するには、[Author from scratch] を選択します。
   - CloudFront のブループリントのリストを表示するには、フィルタフィールドに cloudfront と入力し、Enter キーを選択します。

     使用したい設計図を見つけたら、設計図の名前を選択します。
5. [Basic information] セクションで、以下の値を指定します。

   a. 名前 - 関数の名前を入力します。

   b. ロール – すぐに開始するには、[テンプレートから新しいロールを作成する複数] を選択します。[既存のロールを選択] または [カスタムロールを作成] を選択し、プロンプトに従ってこのセクションの情報を完了することもできます。

   c. ロール名 - ロールの名前を入力します。

   d. ポリシーテンプレート – [基本的な Edge Lambda のアクセス許可] を選択します。
6. ステップ 4 で [Author from scratch] を選択した場合は、ステップ 7 に進んでください。

   ステップ 4 で設計図を選択した場合は、[cloudfront] セクションで、この関数を CloudFront ディストリビューションおよび CloudFront イベントのキャッシュに関連付ける 1 つのトリガーを作

成できます。この時点で [Remove] を選択することをお勧めします。そのため、関数の作成時にトリガーはありません。後でトリガーを追加できます。

> **Tip**
>
> トリガーを追加する前に、関数をテストおよびデバッグすることをお勧めします。代わりにトリガーを追加する場合、関数を作成するとすぐに関数が実行され、世界各地の AWS ロケーションへのレプリケーションが完了し、対応するディストリビューションがデプロイされます。

7. [Create function (関数の作成)] を選択します。

   Lambda は関数 $LATEST とバージョン 1 の 2 つのバージョンを作成します。$LATEST バージョンのみを編集できますが、コンソールに最初はバージョン 1 が表示されます。
8. 関数を編集するには、関数の ARN の下にあるページの上部にある [Version 1] を選択します。次に [Versions] タブで [$LATEST] を選択します。(その関数をそのままにしてから戻った場合、ボタンラベルは [Qualifiers] となります)。
9. [Configuration] タブで、該当する [Code entry type] を選択します。次に、プロンプトに従ってコードを編集またはアップロードします。
10. [ランタイム] で、関数のコードに基づいて値を選択します。
11. [Tags] セクションで、該当するタグを追加します。
12. [Actions] を選択し、[Publish new version] を選択します。
13. 新しいバージョンの関数の説明を入力します。
14. [Publish] を選択します。
15. 関数をテストおよびデバッグします。Lambda コンソールのテストの詳細については、AWS Lambda デベロッパーガイドの「コンソールを使用して Lambda 関数を作成する」で Lambda 関数を手動で呼び出して、結果、ログ、メトリクスを確認するセクションを参照してください。
16. CloudFront イベントに対して関数を実行する準備ができたら、別のバージョンを公開し、トリガーを追加する関数を編集します。詳細については、「Lambda@Edge 関数のトリガーを追加する」を参照してください。

API または AWS CLI を使用して Lambda@Edge を操作する

Lambda および CloudFront API オペレーションを使用して、Lambda@Edge 関数と CloudFront トリガーをプログラムでセットアップすることもできます。詳細については、次のトピックを参照してください。

- AWS LambdaAPI リファレンス
- Amazon CloudFront API リファレンス
- 以下の AWS Command Line Interface (AWS CLI) コマンドを使用することもできます。
  - Lambda create-function
  - CloudFront create-distribution
  - CloudFront create-distribution-with-tags
  - CloudFront update-distribution
- AWS SDK (「SDK とツールキット」セクションを参照)。
- AWS Tools for PowerShell コマンドレットリファレンス

Lambda 関数を変更する

Lambda@Edge 関数を作成したら、Lambda コンソールを使用してそれを変更できます。

> **メモ**
>
> - 元のバージョンのラベルは [$LATEST] です。
> - $LATEST バージョンのみを編集できます。
> - $LATEST バージョンを編集するたびに、新しい番号付きバージョンを公開する必要があります。
> - $LATEST のトリガーを作成することはできません。
> - 新しいバージョンの関数を発行する場合、Lambda は以前のバージョンから新しいバージョンにトリガーを自動的にコピーしません。新しいバージョン用のトリガーを再現する必要があります。
> - CloudFront イベントのトリガーを関数に追加する場合、同じディストリビューション、キャッシュ動作、および同じ関数の以前のバージョン用のイベント用のトリガーが既に存在している場合、Lambda は以前のバージョンからトリガーを削除します。
> - トリガーを追加するなど、CloudFront ディストリビューションを更新した後で、トリガーで指定した関数が機能する前に、変更がエッジロケーションに伝達されるのを待つ必要があります。

Lambda 関数を変更するには (コンソール)

1. AWS Management Console にサインインして AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. ページの上部にあるリージョンのリストで、[米国東部 (バージニア北部)] を選択します。
3. 関数のリストで、関数の名前を選択します。

   デフォルトでは、$LATEST バージョンがコンソールに表示されます。以前のバージョンを表示することはできますが ([Qualifiers] を選択します)、編集できるのは $LATEST のみです。
4. [Code (コード)] タブの [Code entry type (コードの入力タイプ)] で、ブラウザでのコードの編集、.zip ファイルのアップロード、または Amazon S3 からのファイルのアップロードを選択します。
5. [Save] または [Save and test] を選択します。
6. [Actions] を選択し、[Publish new version] を選択します。
7. [Publish new version from $LATEST] ダイアログボックスで、新しいバージョンの説明を入力します。この説明は、自動的に生成されたバージョン番号とともにバージョンのリストに表示されます。
8. [Publish] を選択します。

   新しいバージョンが自動的に最新バージョンになります。バージョン番号はページの左上隅にある [バージョン]に表示されます。
9. [Triggers] タブを選択します。
10. [Add trigger] を選択します。
11. [Add trigger (トリガーの追加)] ダイアログボックスでチェックボックスをオンにし、[CloudFront] を選択します。

    > **Note**
    >
    > 関数の 1 つまたは複数のトリガーを作成済みの場合、CloudFront がデフォルトのサービスになります。

12. Lambda 関数をいつ実行するかを示す、次の値を指定します。

    a. ディストリビューション ID – トリガーの追加先となるディストリビューションの ID を選択します。

    b. キャッシュ動作 – 関数を実行するオブジェクトを指定するキャッシュ動作を選択します。

c. CloudFront イベント – 関数を実行させる CloudFront イベントを選択します。

d. Enable trigger and replicate – このチェックボックスをオンにし、Lambda が関数を AWS リージョン に対してグローバルにレプリケートするようにします。

13. 送信 を選択します。

14. この関数のトリガーをさらに追加するには、ステップ 10 ~ 13 を繰り返します。

Lambda@Edge 関数のトリガーを追加する

Lambda@Edge トリガーは、CloudFront ディストリビューション、キャッシュ動作、および関数を実行させるイベントの 1 つの組み合わせです。関数を実行させる 1 つまたは複数の CloudFront トリガーを指定できます。例えば、ディストリビューション用に設定した特定のキャッシュ動作について、CloudFront がビューワーからリクエストを受け取ったときに関数を実行させるトリガーを作成できます。

> **Tip**
>
> CloudFront ディストリビューションを作成するときは、別のリクエストを受け取ったときに応答する方法を CloudFront に指示する設定を指定します。デフォルトの設定は、ディストリビューションのデフォルトのキャッシュ動作と呼ばれます。特定のファイルタイプのリクエストを受け取る場合など、特定の状況での CloudFront の応答方法を定義する追加のキャッシュ動作を設定できます。詳細については、「キャッシュ動作設定」を参照してください。

Lambda 関数を作成するときに、1 つのトリガーのみを指定できます。Lambda コンソールを使用するか、CloudFront コンソールでディストリビューションを編集するかで、後で同じ関数にさらにトリガーを追加できます。

- Lambda コンソールは、同じ CloudFront ディストリビューションの関数にトリガーを追加する場合に適しています。
- 更新するディストリビューションを見つけやすいため、複数のディストリビューションのトリガーを追加する場合は、CloudFront コンソールが適しています。同時に他の CloudFront 設定を更新することもできます。

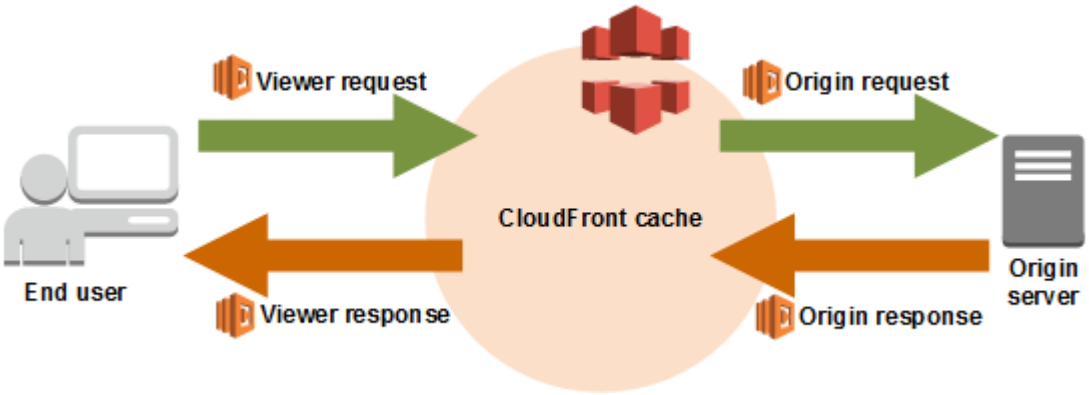
> **Note**
>
> Lambda@Edge をプログラムで操作するには、「API または AWS CLI を使用して Lambda@Edge を操作する」を参照してください。

トピック

- Lambda@Edge 関数をトリガーできる CloudFront イベント
- どの CloudFront イベントを使用して Lambda@Edge 関数をトリガーするかを決定する
- Lambda@Edge 関数にトリガーを追加する (コンソール)

Lambda@Edge 関数をトリガーできる CloudFront イベント

Amazon CloudFront ディストリビューションの各キャッシュ動作に、特定の CloudFront イベントの発生時に Lambda 関数を実行させるトリガー (関連付け) を 4 つまで追加できます。CloudFront トリガーは、次の図に示す、4 つの CloudFront イベントのいずれかに基づくことができます。



Lambda@Edge 関数のトリガーに使用できる CloudFront イベントには、以下のものがあります。

ビューワーリクエスト

CloudFront がビューワーからリクエストを受け取ると、リクエストされたオブジェクトが CloudFront キャッシュにあるかどうかを確認する前に関数が実行されます。

オリジンリクエスト

CloudFront がリクエストをオリジンに転送したときにのみ、関数が実行されます。リクエストされたオブジェクトが CloudFront キャッシュ内にある場合、関数は実行されません。

オリジンレスポンス

CloudFront がオリジンからのレスポンスを受け取った後、レスポンス内のオブジェクトをキャッシュする前に関数が実行されます。関数は、オリジンからエラーが返された場合でも実行されることに注意してください。

次の場合には関数は実行されません。

- リクエストされたファイルが CloudFront キャッシュ内にあり、その有効期限が切れていない場合。
- オリジンリクエストイベントによってトリガーされた関数からレスポンスが生成された場合。

ビューワーレスポンス

リクエストされたファイルがビューワーに返される前に関数が実行されます。ファイルが CloudFront キャッシュ内に既に存在するかどうかに関係なく、関数が実行されることに注意してください。

次の場合には関数は実行されません。

- オリジンが HTTP ステータスコードとして 400 以上を返した場合。
- カスタムエラーページが返された場合。
- ビューワーリクエストイベントによってトリガーされた関数からレスポンスが生成された場合。
- CloudFront で HTTP リクエストが自動的に HTTPS にリダイレクトされる場合 ([ビューワープロトコルポリシー] の値が [Redirect HTTP to HTTPS] の場合)

同じキャッシュ動作に複数のトリガーを追加する場合、各トリガーに対して同じ関数を実行することも、異なる関数を実行することもできます。また、複数のディストリビューションに同じ関数を関連付けることもできます。

> **Note**
>
> CloudFront イベントによって Lambda 関数の実行がトリガーされると、その関数が終了するまで CloudFront は続行できません。例えば、CloudFront ビューワーリクエストイベントによって Lambda 関数がトリガーされると、その Lambda 関数が実行を終了するまで、CloudFront からビューワーにレスポンスは返されず、リクエストはオリジンに転送されません。つまり、Lambda 関数をトリガーするリクエストごとにリクエストのレイテンシーが長くなるため、関数をできるだけ速く実行する必要があります。

どの CloudFront イベントを使用して Lambda@Edge 関数をトリガーするかを決定する

Lambda 関数をトリガーするために使用する CloudFront イベントを決定する際には、次の点を考慮してください。

Lambda 関数によって変更されたオブジェクトを CloudFront でキャッシュするかどうか

Lambda 関数によって変更されたオブジェクトを CloudFront でキャッシュして、次にそのオブジェクトがリクエストされたときにエッジロケーションから提供できるようにする場合は、オリジンリクエストイベントかオリジンレスポンスイベントを使用します。これにより、オリジンの負荷と以降のリクエストのレイテンシーが軽減され、以降のリクエストで Lambda@Edge を呼び出すコストが削減されます。

例えば、オリジンから返されたオブジェクトのヘッダーを追加、削除、または変更する場合に、その結果を CloudFront でキャッシュするには、オリジンレスポンスイベントを使用します。

すべてのリクエストに対して関数を実行するかどうか

CloudFront が受信したディストリビューションのすべてのリクエストに対して関数を実行する場合は、ビューワーリクエストイベントかビューワーレスポンスイベントを使用します。オリジンリクエストイベントとオリジンレスポンスイベントは、リクエストされたオブジェクトがエッジロケーションにキャッシュされておらず、CloudFront がリクエストをオリジンに転送する場合にだけ発生します。

関数でキャッシュキーを変更するかどうか

キャッシュ条件として使用している値を関数で変更する場合はビューワーリクエストイベントを使用します。たとえば、関数で URL を変更してパスに言語の省略形を含める場合 (ユーザーがドロップダウンリストから言語を選択した場合など) は、ビューワーリクエストイベントを使用します。

- ビューワーリクエストの URL - https://example.com/en/index.html
- リクエストがドイツの IP アドレスから送られてきた場合の URL - https://example.com/de/index.html

Cookie またはリクエストヘッダーをキャッシュ条件として使用している場合もビューワーリクエストイベントを使用します。

> **Note**
>
> 関数で Cookie またはヘッダーを変更する場合は、リクエストの該当部分をオリジンに転送するように CloudFront を設定します。詳細については、以下のトピックを参照してください。
>
> - Cookie に基づいてコンテンツをキャッシュする
> - リクエストヘッダーに基づいてコンテンツをキャッシュする

関数がオリジンからのレスポンスに影響するかどうか

関数でリクエストに加える変更がオリジンからのレスポンスに影響する場合は、オリジンリクエストイベントを使用します。通常、ほとんどのビューワーリクエストイベントはオリジンに転送されません。 CloudFront がエッジキャッシュにすでに存在するオブジェクトを使用してリクエストに応答します。オリジンリクエストイベントに基づく関数でリクエストを変更すると、変更されたオリジンリクエストに対するレスポンスが CloudFront でキャッシュされます。

Lambda@Edge 関数にトリガーを追加する (コンソール)

AWS Lambda コンソールまたは Amazon CloudFront コンソールのいずれかを使用して、Lambda@Edge 関数のトリガーを追加できます。

> **Important**
>
> トリガーは、関数の番号付きバージョン ($LATEST ではなく) に対してのみ作成できます。

Lambda console

Lambda@Edge 関数にトリガーを追加するには (コンソール)

1. AWS Management Console にサインインして AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. ページの上部にあるリージョンのリストで、[米国東部 (バージニア北部)] を選択します。
3. [Functions] ページで、トリガーを追加する関数の名前を選択します。
4. [関数の概要] ページで、[バージョン] タブを選択します。
5. トリガーを追加するバージョンを選択します。

バージョンを選択すると、ボタンの名前が [Version: $LATEST] または [Version: バージョン番号] に変わります。

6. [Triggers] タブを選択します。
7. [Add trigger] を選択します。
8. [トリガー設定] で、[ソースを選択] を選択し、**`cloudfront`** と入力し、CloudFront を選択します。

> **Note**
>
> 1 つまたは複数のトリガーを作成済みの場合、CloudFront がデフォルトのサービスになります。

9. Lambda 関数をいつ実行するかを示す、次の値を指定します。
   a. ディストリビューション – トリガーを追加するディストリビューションを選択します。
   b. キャッシュ動作 – 関数を実行するオブジェクトを指定するキャッシュ動作を選択します。

   > **Note**
   >
   > キャッシュ動作に * を指定すると、Lambda 関数はデフォルトのキャッシュ動作にデプロイされます。

   c. CloudFront イベント – 関数を実行させる CloudFront イベントを選択します。
   d. 本文を含める – 関数のリクエストボディにアクセスするには、このチェックボックスをオンにします。
   e. Lambda@Edge へのデプロイを確認 – このチェックボックスをオンにして、AWS Lambda が関数を AWS リージョン にグローバルにレプリケートするようにします。
10. 追加 を選択します。

この関数は、更新された CloudFront ディストリビューションがデプロイされたときに、指定された CloudFront イベントのリクエストの処理を開始します。ディストリビューションがデプロイされているかどうかを確認するには、ナビゲーションペインで [Distributions] を選択します。ディストリビューションをデプロイすると、ディストリビューションの [ステータス] 列の値が、[デプロイ中] からデプロイの日時に変わります。

CloudFront console

CloudFront イベントのトリガーを Lambda 関数に追加するには

1. トリガーを追加する Lambda 関数の ARN を取得します。

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

   b. ページの上部にあるリージョンのリストで、[米国東部 (バージニア北部)] を選択します。

   c. 関数のリストで、トリガーを追加する関数の名前を選択します。

   d. [関数の概要] ページで [バージョン] タブを選択し、トリガーを追加する先の番号付きバージョンを選択します。

   e. [ARN をコピー] ボタンを選択して、ARN をクリップボードにコピーします。Lambda 関数の ARN は次のようになります。

      ```
      arn:aws:lambda:us-east-1:123456789012:function:TestFunction:2
      ```

      末尾の番号 (この例では 2) は関数のバージョン番号です。

2. で CloudFront コンソールを開きますhttps://console.aws.amazon.com/cloudfront/v4/home

3. ディストリビューションのリストで、トリガーを追加するディストリビューションの ID を選択します。

4. [Behaviors] タブを選択します。

5. トリガーを追加する先のキャッシュビヘイビアを選択し、[編集] を選択します。

6. [関数の関連付け] で、[関数タイプ] リストから [Lambda@Edge] を選択し、ビューワーリクエスト、ビューワーレスポンス、オリジンリクエスト、またはオリジンレスポンスに対して関数をいつ実行するかを選択します。

   詳細については、「どの CloudFront イベントを使用して Lambda@Edge 関数をトリガーするかを決定する」を参照してください。

7. [関数 ARN/名前] テキストボックスに、選択したイベントの発生時に実行する Lambda 関数の ARN を貼り付けます。これは Lambda コンソールからコピーした値です。

8. 関数のリクエスト本文にアクセスする場合は、[本文を含める] を選択します。

   リクエスト本文を置き換えるだけの場合は、このオプションを選択する必要はありません。

9. 他のイベントタイプで同じ関数を実行するには、ステップ 6 と 7 を繰り返します。

10. [Save changes] (変更の保存) をクリックします。
11. このディストリビューションの他のキャッシュビヘイビアにトリガーを追加するには、ステップ 5～10 を繰り返します。

    この関数は、更新された CloudFront ディストリビューションがデプロイされたときに、指定された CloudFront イベントのリクエストの処理を開始します。ディストリビューションがデプロイされているかどうかを確認するには、ナビゲーションペインで [Distributions] を選択します。ディストリビューションをデプロイすると、ディストリビューションの [ステータス] 列の値が、[デプロイ中] からデプロイの日時に変わります。

Lambda@Edge 関数をテストおよびデバッグする

このトピックでは、Lambda@Edge 関数をテストおよびデバッグするための戦略を説明するセクションを示します。Lambda@Edge 関数コードのスタンドアロンをテストすること、目的のタスクの完了を確認すること、統合のテストを行うこと、CloudFront で関数が正しく機能しているか確認することは重要です。

統合テスト中または関数がデプロイされた後に、HTTP 5xx エラーなどの CloudFront エラーのデバッグが必要になることがあります。エラーは、Lambda 関数から返される無効なレスポンス、関数がトリガーされるときの実行時のエラー、または Lambda サービスによる実行スロットリングが原因のエラーの可能性があります。このトピックのセクションでは、どのタイプの障害が問題であるかを判別するための戦略、そしてその問題を解決するためのステップを共有します。

> **Note**
>
> エラーをトラブルシューティングするときに CloudWatch ログファイルまたはメトリクスを確認する場合は、関数が実行される場所に最も近い AWS リージョン に表示または保存されていることに注意してください。したがって、例えば英国のユーザーがいるウェブサイトまたはウェブアプリケーションで、ディストリビューションに関連する Lambda 関数がある場合は、リージョンを変更してロンドン AWS リージョン の CloudWatch メトリクスまたはログファイルを表示する必要があります。詳細については、「the section called “Lambda@Edge リージョンを判断する”」を参照してください。

トピック

- Lambda@Edge 関数をテストする
- CloudFront での Lambda@Edge 関数エラーを識別する

- 無効な Lambda@Edge 関数レスポンス (検証エラー) のトラブルシューティング
- Lambda@Edge 関数実行エラーをトラブルシューティングする
- Lambda@Edge リージョンを判断する
- アカウントがログを CloudWatch にプッシュするかどうかを判断する

Lambda@Edge 関数をテストする

Lambda 関数をテストするには、スタンドアロンテストと統合テストの 2 つのステップがあります。

スタンドアロン機能のテスト

CloudFront に Lambda 関数を追加する前に、Lambda コンソールでテスト機能を使用するか他の方法を使用して、必ず最初に機能をテストしてください。Lambda コンソールのテストの詳細については、AWS Lambda デベロッパーガイドの「コンソールを使用して Lambda 関数を作成する」で Lambda 関数を手動で呼び出して、結果、ログ、メトリクスを確認するセクションを参照してください。

CloudFront での関数のオペレーションのテスト

統合テストを完了することが重要です。ここで、関数はディストリビューションに関連付けられ、CloudFront イベントに基づいて実行されます。関数が正しいイベントに対してトリガーされることを確認し、CloudFront に対して有効で正しいレスポンスを返します。例えば、イベント構造が正しいこと、有効なヘッダーだけが含まれていることなどを確認します。

Lambda コンソールの関数で統合テストを繰り返す場合、コードを変更する際や関数を呼び出す CloudFront トリガーを変更する際は、Lambda@Edge チュートリアルのステップを参照しください。例えば、チュートリアルの「ステップ 4: 関数を実行する CloudFront トリガーを追加する」のステップで説明しているように、関数の番号付きバージョンを操作していることを確認します。

変更を加えた場合やデプロイした場合、更新した関数と CloudFront トリガーがすべてのリージョンにレプリケートするまで数分かかることに注意してください。通常、これには数分かかりますが、最大で 15 分かかる場合があります。

レプリケーションが終了したかどうかを確認するには、CloudFront コンソールに移動し、ディストリビューションを表示します。

レプリケーションのデプロイが完了したかどうかを確認するには

1. CloudFront コンソール (https://console.aws.amazon.com/cloudfront/v4/home) を開きます。

2. ディストリビューション名を選択します。
3. ディストリビューションのステータスが [進行中] から [デプロイ済み] に戻ったことを確認します。この場合、関数はレプリケートされたことを意味します。続いて、次のセクションのステップに従って関数が機能することを確認します。

コンソールでのテストでは関数のロジックのみを検証します。また、Lambda@Edge に固有のサービスクォータ (以前は制限と呼ばれていました) は適用されないことに注意してください。

CloudFront での Lambda@Edge 関数エラーを識別する

関数のロジックが正常に機能することを確認した後でも、CloudFront での関数の実行時に HTTP 5xx エラーが発生することがあります。HTTP 5xx エラーはさまざまな理由で返される可能性があります。これには、Lambda 関数エラーやその他の CloudFront の問題が含まれる場合があります。

- Lambda@Edge 関数を使用している場合は、CloudFront コンソールのグラフを使用してエラーの原因を突き止め、それを修正することができます。例えば、HTTP 5xx エラーの原因が CloudFront によるものか、Lambda 関数によるものかを確認し、特定の関数については関連するログファイルを表示して問題を調査できます。
- HTTP エラー全般を CloudFront でトラブルシューティングするには、オリジンからのエラーレスポンスのトラブルシューティング のトピックのトラブルシューティング手順を参照してください。

CloudFront で Lambda@Edge 関数エラーが発生する原因

Lambda 関数が HTTP 5xx エラーの原因となる可能性がある理由はいくつかあります。実行するトラブルシューティングステップはエラーのタイプによって異なります。エラーは次のように分類されます。

Lambda 関数実行エラー

関数に未処理の例外があるか、コードにエラーがあって CloudFront が Lambda からレスポンスを得られない場合は、実行エラーが発生します。たとえば、コードにコールバック (エラー) が含まれている場合です。詳細については、「AWS Lambda デベロッパーガイド」の「Lambda 関数のエラー」を参照してください。

無効な Lambda 関数のレスポンスが CloudFront に返される

関数の実行後、CloudFront が Lambda からレスポンスを受け取ります。レスポンスのオブジェクト構造が Lambda@Edge イベント構造 に従わない場合、またはレスポンスに無効なヘッダーや他の無効なフィールドが含まれている場合、エラーが返されます。

CloudFront での実行は、Lambda サービスのクォータ (以前は制限と呼ばれていました) のために調整されます。

Lambda サービスは各リージョンでの実行を制限し、クォータに達するとエラーが返されます。

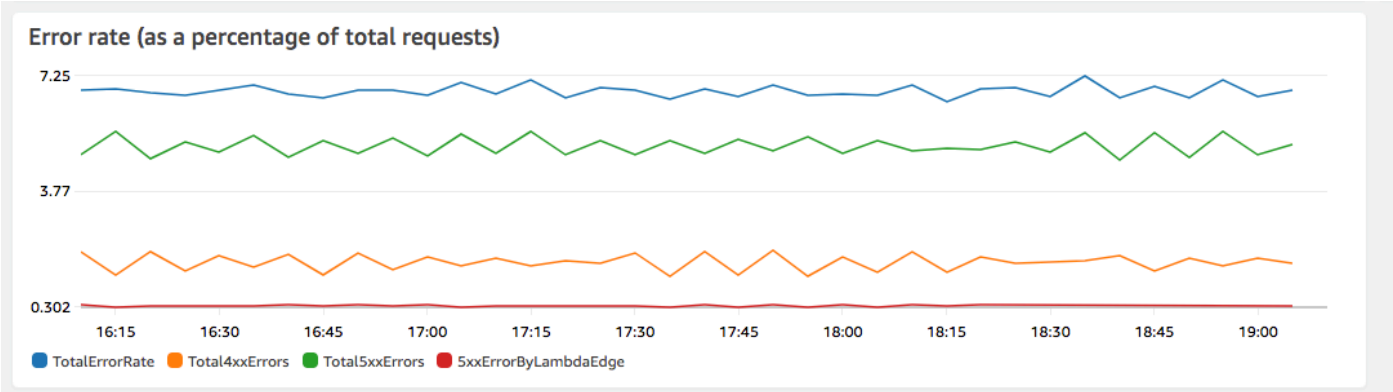
障害のタイプを判断する方法

デバッグするときにどこに焦点を合わせて CloudFront から返されたエラーを解決するかを決めるのに役立つように、なぜ CloudFront が HTTP エラーを返しているのかを識別することは役立ちます。これを開始するには、AWS Management Console で CloudFront コンソールの [Monitoring] (モニタリング) セクションにあるグラフを使うことができます。CloudFront コンソールの [Monitoring (モニタリング)] セクションでのグラフ表示の詳細については、「Amazon CloudWatch による CloudFront メトリクスのモニタリング」を参照してください。

次のグラフは、エラーが発生源によって返されたのか Lambda 関数によって返されたのかを追跡し、Lambda 関数からのエラーである場合に問題の種類を絞り込む場合に特に役立ちます。

エラー率グラフ

各ディストリビューションの [Overview] タブに表示できるグラフの1つが、[Error rates] グラフです。このグラフは、ディストリビューションに対するすべてのリクエストに対するエラーの割合をパーセンテージで表示します。グラフは、Lambda 関数の合計エラー率、合計 4xx エラー、合計 5xx エラー、合計 5xx エラーを示しています。エラーの種類と量に基づいて、原因を調査してトラブルシューティングするための手順を実行できます。



- Lambda エラーが表示された場合は、関数が返す特定の種類のエラーを調べることで、さらに詳しく調べることができます。[Lambda@Edge errors] タブには、特定の関数に関する問題を特定するのに役立つように、関数エラーをタイプ別に分類したグラフが含まれています。
- CloudFront エラーが表示された場合は、トラブルシューティングを行い、オリジンエラーを修正したり、CloudFront 設定を変更したりすることができます。詳細については、「オリジンからのエラーレスポンスのトラブルシューティング」を参照してください。

Execution エラーと無効な関数レスポンスグラフ

[Lambda@Edge errors] タブには、特定のディストリビューションに対する Lambda@Edge エラーをタイプ別に分類したグラフが含まれています。例えば、1 つのグラフに AWS リージョン別の実行エラーがすべて表示されます。

問題のトラブルシューティングを容易にするために、地域別に特定の関数のログファイルを開いて調べることで、特定の問題を探すことができます。

リージョン別に特定の関数のログファイルを表示するには

1. [Lambda@Edge エラー] タブの [関連する Lambda@Edge 関数] で、関数名を選択し、[メトリクスの表示] を選択します。
2. 次に、関数名のページの右上隅で、[関数ログの表示] を選択し、リージョンを選択します。

   例えば、米国西部 (オレゴン) リージョンの [エラー] グラフに問題が表示される場合は、ドロップダウンリストからそのリージョンを選択します。これにより、Amazon CloudWatch コンソールが開きます。
3. そのリージョンの CloudWatch コンソールの [ログストリーム] で、ログストリームを選択して関数のイベントを表示します。

さらに、トラブルシューティングとエラーの修正に関する推奨事項については、この章の次のセクションを参照してください。

スロットルグラフ

[Lambda@Edge errors] タブには、[Throttles] グラフも含まれます。場合によっては、リージョンの同時実行性のクォータに達すると、Lambda サービスがリージョンごとに関数呼び出しを調整します。制限の超過エラーが表示される場合は、Lambda サービスがリージョンの実行に課すクォータに関数が達しています。クォータの増加をリクエストする方法など、詳細については、「Lambda@Edge のクォータ」を参照してください。

HTTP エラーのトラブルシューティングでこの情報を使用する方法の例については、「AWS でコンテンツ配信をデバッグするための 4 つのステップ」を参照してください。

無効な Lambda@Edge 関数レスポンス (検証エラー) のトラブルシューティング

問題が Lambda 検証エラーであると特定した場合は、Lambda 関数が CloudFront に無効なレスポンスを返していることを意味します。このセクションのガイダンスに従い、関数を確認し、レスポンスが CloudFront 要件に従っていることを確認する手順を実行します。

CloudFront は、次の 2 つの方法で Lambda 関数からのレスポンスを検証します。

- Lambda レスポンスは、必要なオブジェクト構造に従う必要があります。不正なオブジェクト構造の例には次のようなものがあります。解析できない JSON、必須フィールドの欠落、レスポンスの無効なオブジェクト。詳細については、「Lambda@Edge イベント構造」を参照してください。
- レスポンスには有効なオブジェクト値のみを含める必要があります。レスポンスに有効なオブジェクトが含まれるがサポートされていない値がある場合、エラーが発生します。例には、許可されていない、または読み取り専用のヘッダーの追加または更新 (「エッジ関数に対する制限」を参照)、ボディサイズの上限の超過 (Lambda@Edge エラー トピックの「生成されるレスポンスのサイズに対する制限」を参照)、および無効な文字または値 (「Lambda@Edge イベント構造」を参照) などがあります。

Lambda が CloudFront に無効なレスポンスを返すと、Lambda 関数が実行されるリージョンで CloudFront が CloudWatch にプッシュするログファイルに、エラーメッセージが書き込まれます。これは、無効なレスポンスがあるときにログファイルを CloudWatch に送信するデフォルトの動作

です。ただし、機能をリリースする前に Lambda 関数を CloudFront と関連付けると、関数に対して有効にならない可能性があります。詳細については、このトピックの後半の「アカウントがログを CloudWatch にプッシュするかどうかを判断する」を参照してください。

CloudFront は、関数を実行した場所に対応するリージョンで、ディストリビューションに関連するロググループにログファイルをプッシュします。ロググループの形式は `/aws/cloudfront/LambdaEdge/DistributionId` です。ここで *`DistributionId`* はディストリビューションの ID です。CloudWatch ログファイルがあるリージョンを決定するには、このトピックの後半の「Lambda@Edge のリージョンの判別」を参照してください。

再現可能なエラーの場合、エラーになる新しいリクエストを作成し、障害のある CloudFront レスポンス (`X-Amz-Cf-Id` ヘッダー) でリクエスト ID を見つけて、ログファイル内の 1 つの障害を特定します。ログファイルのエントリには、エラーが返される理由を特定するのに役立つ情報が含まれます。また、対応する Lambda リクエスト ID もリスト表示されるため、1 つのリクエストのコンテキストでの根本原因を分析することもできます。

エラーが断続的な場合は、CloudFront アクセスログを使用して障害が発生したリクエストのリクエスト ID を見つけ、対応するエラーメッセージの CloudWatch Logs を検索します。詳細については、前のセクションの「Determining the Type of Failure」を参照してください。

Lambda@Edge 関数実行エラーをトラブルシューティングする

Lambda 実行エラーが問題である場合は、Lambda 関数のログ記録ステートメントを作成しておくと、CloudWatch での関数の実行をモニタリングする CloudFront ログファイルにメッセージを書き込み、正常に機能しているかどうかを判断するのに役立ちます。その後、これらのステートメントを CloudWatch ログファイルで検索して、関数が正常に機能していることを確認できます。

> **Note**
>
> Lambda@Edge 関数を変更していない場合でも、Lambda 関数の実行環境を更新すると、この関数に影響を与え、実行エラーが発生する可能性があります。テストおよび新しいバージョンへの移行の詳細については、「Upcoming updates to the AWS Lambda and AWS Lambda@Edge execution environment」を参照してください。

Lambda@Edge リージョンを判断する

Lambda@Edge 関数がトラフィックを受信しているリージョンを確認するには、AWS Management Console のCloudFront コンソールでその関数のメトリクスを表示します。メトリクスは AWS リー

ジョンごとに表示されます。同じページで、リージョンを選択してそのリージョンのログファイルを表示し、問題を調査することができます。CloudFront が Lambda 関数を実行したときに作成されたログファイルを表示するには、正しい AWS リージョンで CloudWatch ログファイルを確認する必要があります。

CloudFront コンソールの [Monitoring (モニタリング)] セクションでのグラフ表示の詳細については、「Amazon CloudWatch による CloudFront メトリクスのモニタリング」を参照してください。

アカウントがログを CloudWatch にプッシュするかどうかを判断する

デフォルトでは、CloudFront によって無効な Lambda 関数レスポンスのログ記録が有効になり、Lambda@Edge 用のサービスにリンクされたロール のいずれかを使用してログファイルが CloudWatch にプッシュされます。無効な Lambda 関数レスポンスのログ機能がリリースされた前に CloudFront に追加した Lambda@Edge 関数がある場合は、CloudFront トリガーを追加するなど、Lambda@Edge 設定を次に更新するときにログ記録が有効になります。

次を実行して、アカウントでログファイルを CloudWatch にプッシュすることが有効になっているか確認できます。

- ログが CloudWatch に表示されているか確認する。必ず、Lambda@Edge 関数が実行されたリージョンを確認します。詳細については、「 Lambda@Edge リージョンを判断する」を参照してください。
- 関連するサービスリンクロールが IAM のアカウントに存在するかどうかを確認する。これを行うには、https://console.aws.amazon.com/iam/ で IAM コンソールを開いてから [ロール] を選択して、アカウントのサービスリンクロールのリストを表示します。次のロールを探してください。`AWSServiceRoleForCloudFrontLogger`

Lambda@Edge 関数とレプリカを削除する

Lambda@Edge 関数を削除できるのは、関数のレプリカが CloudFront によって削除された場合のみです。Lambda 関数のレプリカは、次のような状況では自動的に削除されます。

- すべての CloudFront ディストリビューションから関数の最後の関連付けを削除した後。複数のディストリビューションで関数が使用されている場合、最後のディストリビューションから関数の関連付けを削除した後にのみ、レプリカが削除されます。
- 関数が関連付けられた最後のディストリビューションを削除した後。

レプリカは通常、数時間以内に削除されます。Lambda@Edge 関数のレプリカを手動で削除することはできません。これにより、まだ使用中のレプリカが削除され、エラーが発生する状況を防ぐことができます。

> **Warning**
>
> CloudFront の外部で Lambda@Edge 関数のレプリカを使用するアプリケーションを構築しないでください。これらのレプリカは、ディストリビューションとの関連付けが削除されるか、ディストリビューション自体が削除されると削除されます。外部のアプリケーションが依存するレプリカが警告なしに削除されて、エラーが発生することがあります。

CloudFront ディストリビューションから Lambda@Edge 関数の関連付けを削除するには (コンソール)

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. 削除する Lambda@Edge 関数の関連付けがあるディストリビューションの ID を選択します。
3. [Behaviors] タブを選択します。
4. 削除する Lambda@Edge 関数の関連付けがあるキャッシュビヘイビアを選択し、[編集] を選択します。
5. Lambda@Edge 関数の関連付けを削除するには、[関数の関連付け]、[関数タイプ]、[関連付けなし] の順に選択します。
6. [Save changes] (変更の保存) をクリックします。

CloudFront ディストリビューションから Lambda@Edge 関数の関連付けを削除した後、必要に応じて AWS Lambda から Lambda 関数または関数バージョンを削除することもできます。関数の関連付けを削除したら、Lambda@Edge 関数のレプリカをクリーンアップできるようになるまで数時間待ちます。その後、Lambda コンソール、AWS CLI、Lambda API、または AWS SDK を使用して関数を削除できます。

特定のバージョンの Lambda 関数が、どの CloudFront ディストリビューションにも関連付けられていない場合は、このバージョンを削除することもできます。Lambda 関数バージョンのすべての関連付けを削除したら、数時間待ちます。その後に関数バージョンを削除できます。

Lambda@Edge イベント構造

以下のトピックでは、CloudFront がトリガーされたときに Lambda@Edge 関数に渡すリクエストおよびレスポンスイベントオブジェクトについて説明します。

トピック

動的オリジン選択

キャッシュ動作でパスパターンを使用すると、リクエストされたオブジェクトのパスと名前 (`images/*.jpg` など) に基づいて、リクエストをオリジンにルーティングできます。Lambda@Edge を使用すると、リクエストヘッダーの値など他のプロパティに基づいても、リクエストをオリジンにルーティングできます。

この動的オリジン選択が便利な状況がいくつかあります。たとえば、グローバルな負荷分散に役立つように、地理的に異なるリージョンのオリジンにリクエストを分散させる場合です。あるいは、機能 (ボット処理、SEO 最適化、認証など) が異なるさまざまなオリジンに、リクエストを選択的にルーティングする場合です。この機能の使用方法を示すコードサンプルについては、「コンテンツベースの動的オリジンの選択 - 例」を参照してください。

CloudFront のオリジンリクエストイベントで、イベント構造の `origin` オブジェクトには、パスパターンに基づいてリクエストがルーティングされるオリジンに関する情報が含まれます。リクエストを別のオリジンにルーティングするように、`origin` オブジェクトの値を更新できます。`origin` オブジェクトを更新するときに、ディストリビューションのオリジンを定義する必要はありません。Amazon S3 オリジンオブジェクトをカスタムオリジンオブジェクトに置き換えたり、その逆にすることもできます。ただし、カスタムオリジンと Amazon S3 オリジンのどちらか (両方は不可) を通じて、リクエストごとに 1 つのオリジンしか指定できません。

リクエストイベント

以下のトピックでは、ビューワーおよびオリジンリクエストイベントの Lambda 関数に CloudFront が渡すオブジェクトの構造を示します。これらの例は、本文のない `GET` リクエストを示しています。例に続いて、ビューワーとオリジンリクエストイベントで使用可能なすべてのフィールドのリストを示します。

トピック

ビューワーリクエストの例

次の例は、ビューワーリクエストイベントオブジェクトを示しています。

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "viewer-request",
          "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "host": [
              {
                "key": "Host",
                "value": "d111111abcdef8.cloudfront.net"
              }
            ],
            "user-agent": [
              {
                "key": "User-Agent",
                "value": "curl/7.66.0"
              }
            ],
            "accept": [
              {
                "key": "accept",
                "value": "*/*"
              }
            ]
          },
          "method": "GET",
```

```
          "querystring": "",
          "uri": "/"
        }
      }
    }
  ]
}
```

オリジンリクエストの例

次の例は、オリジンリクエストイベントオブジェクトを示しています。

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "origin-request",
          "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "x-forwarded-for": [
              {
                "key": "X-Forwarded-For",
                "value": "203.0.113.178"
              }
            ],
            "user-agent": [
              {
                "key": "User-Agent",
                "value": "Amazon CloudFront"
              }
            ],
            "via": [
              {
                "key": "Via",
                "value": "2.0 2afae0d44e2540f472c0635ab62c232b.cloudfront.net
 (CloudFront)"
              }
            ],
```

```
            "host": [
              {
                "key": "Host",
                "value": "example.org"
              }
            ],
            "cache-control": [
              {
                "key": "Cache-Control",
                "value": "no-cache"
              }
            ]
          },
          "method": "GET",
          "origin": {
            "custom": {
              "customHeaders": {},
              "domainName": "example.org",
              "keepaliveTimeout": 5,
              "path": "",
              "port": 443,
              "protocol": "https",
              "readTimeout": 30,
              "sslProtocols": [
                "TLSv1",
                "TLSv1.1",
                "TLSv1.2"
              ]
            }
          },
          "querystring": "",
          "uri": "/"
        }
      }
    }
  ]
}
```

リクエストイベントフィールド

リクエストイベントオブジェクトデータは、config (Records.cf.config) と request (Records.cf.request) の 2 つのサブオブジェクトに含まれています。次のリストは、各サブオブジェクトのフィールドを示しています。

設定オブジェクトのフィールド

次のリストでは、config オブジェクト (Records.cf.config) のフィールドについて説明します。

distributionDomainName (読み取り専用)

リクエストに関連付けられているディストリビューションのドメイン名。

distributionID (読み取り専用)

リクエストに関連付けられているディストリビューションの ID。

eventType (読み取り専用)

リクエストに関連付けられているトリガーのタイプ (viewer-request または origin-request)。

requestId (読み取り専用)

ビューワーから CloudFront へのリクエストを一意に識別する暗号化された文字列。requestId の値は CloudFront アクセスログにも x-edge-request-id として表示されます。詳細については、標準ログ (アクセスログ) の設定および使用および標準ログファイルフィールドを参照してください。

リクエストオブジェクトのフィールド

次のリストでは、request オブジェクト (Records.cf.request) のフィールドについて説明します。

clientIp (読み取り専用)

リクエストを行ったビューワーの IP アドレス。ビューワーが HTTP プロキシまたはロードバランサーを使用してリクエストを送った場合、この値はプロキシまたはロードバランサーの IP アドレスです。

headers (読み書き)

リクエストのヘッダー。次の点に注意してください。

- headers オブジェクトのキーは標準の HTTP ヘッダー名を小文字にしたものです。小文字のキーを使用して、大文字と小文字を区別せずにヘッダー値にアクセスできます。

- 各ヘッダーオブジェクト (headers["accept"]、headers["host"] など) はキーと値のペアの配列です。返されたヘッダーの配列には、リクエストの値ごとに 1 つのキーと値のペアが含まれます。
- key には、HTTP リクエストに表示されるヘッダーの大文字と小文字を区別する名前が含まれます (Host、User-Agent、X-Forwarded-For など)。
- value には、HTTP リクエストに表示されるヘッダー値が含まれます。
- Lambda 関数がリクエストヘッダーを追加または変更し、ヘッダー key フィールドを含めない場合、Lambda@Edge は指定したヘッダー名を使用してヘッダー key を自動的に挿入します。ヘッダー名をどのようにフォーマットしたかにかかわらず、自動的に挿入されるヘッダーキーの各パートは、先頭が大文字になり、ハイフン (-) で区切られます。

  たとえば、ヘッダー key なしで次のようなヘッダーを追加できます。

  ```
  "user-agent": [
    {
      "value": "ExampleCustomUserAgent/1.X.0"
    }
  ]
  ```

  この例では、Lambda@Edge は "key": "User-Agent" を自動的に挿入します。

ヘッダー使用の制限の詳細については、「エッジ関数に対する制限」を参照してください。

method (読み取り専用)

リクエストの HTTP メソッド。

querystring (読み書き)

リクエスト内のクエリ文字列 (存在する場合)。リクエストにクエリ文字列が含まれていない場合でも、イベントオブジェクトには querystring が含まれ、値が空になります。クエリ文字列の詳細については、「クエリ文字列パラメータに基づいてコンテンツをキャッシュする」を参照してください。

uri (読み書き)

リクエストされたオブジェクトの相対パス。Lambda 関数が uri 値を変更する場合、次の点に注意してください。

- 新しい uri 値は、スラッシュ (/) で始める必要があります。
- 関数で uri 値を変更すると、ビューワーがリクエストしているオブジェクトが変更されます。

- 関数で uri 値を変更しても、リクエストのキャッシュ動作や送信先オリジンは変わりません。

body (読み書き)

HTTP リクエストの本文。body 構造には、次のフィールドを含めることができます。

inputTruncated (読み取り専用)

本文が Lambda@Edge で切り捨てられたかどうかを示すブーリアン型フラグ。詳細については、「Include Body オプションがあるリクエストボディに対する制限」を参照してください。

action (読み書き)

本文で実行する予定のアクション。action のオプションは次のとおりです。

- read-only: これがデフォルト値です。Lambda 関数からレスポンスを返す際に、action が読み取り専用の場合、Lambda@Edge は encoding または data への変更をすべて無視します。
- replace: オリジンに送信される本文を置き換えるときに指定します。

encoding (読み書き)

本文のエンコード。Lambda@Edge が Lambda 関数に本文を公開すると、まず本文を base64-encoding に変換します。本文を置き換える action として replace を選択した場合、base64 (デフォルト) または text エンコードを使用することもできます。encoding を base64 と指定したが本文が有効な base64 でない場合、CloudFront はエラーを返します。

data (読み書き)

リクエストボディのコンテンツ。

origin (読み書き) (オリジンイベントのみ)

リクエストの送信先のオリジン。origin 構造には、オリジンが 1 つだけ含まれていなければなりません。オリジンはカスタムオリジンでも Amazon S3 オリジンでも構いません。オリジン構造には、次のフィールドを含めることができます。

customHeaders (読み取り/書き込み) (カスタムおよび Amazon S3 オリジン)

各カスタムヘッダーの名前と値のペアを指定することで、カスタムヘッダーをリクエストに含めることができます。許可されていないヘッダーを追加することはできず、同じ名前のヘッダーを Records.cf.request.headers に含めることもできません。リクエストヘッダーに関する注意事項は、カスタムヘッダーにも適用されます。詳細については、CloudFront で

オリジンリクエストに追加できないカスタムヘッダーおよびエッジ関数に対する制限を参照してください。

domainName (読み取り/書き込み) (カスタムおよび Amazon S3 オリジン)

オリジンのドメイン名。ドメイン名を空にすることはできません。

- カスタムオリジンの場合 – DNS ドメイン名を指定します (`www.example.com` など)。ドメイン名にコロン (:) を含めることはできません。また、IP アドレスにすることはできません。ドメイン名の最大長は 253 文字です。
- Amazon S3 オリジンの場合 - Amazon S3 バケットの DNS ドメイン名を指定します (`awsexamplebucket.s3.eu-west-1.amazonaws.com` など)。この名前は最大 128 文字で、すべて小文字であることが必要です。

path (読み取り/書き込み) (カスタムおよび Amazon S3 オリジン)

リクエストがコンテンツを検索するサーバーのディレクトリパス。パスは、先頭をスラッシュ (/) にする必要があります。末尾をスラッシュ (/) にすることはできません (例えば、末尾が `example-path/` は不可です)。カスタムオリジンの場合のみ、パスは URL エンコードされ、最大長は 255 文字にする必要があります。

keepaliveTimeout (読み書き) (カスタムオリジンのみ)

CloudFront がレスポンスの最後のパケットを受け取ってからオリジンへの接続を維持しようとする期間 (秒)。この値には、1 ~ 60 の範囲の数値を指定する必要があります。

port (読み書き) (カスタムオリジンのみ)

カスタムオリジンでの CloudFront の接続先ポート。ポートは 80 または 443 であるか、1024 ～ 65535 の範囲の数値であることが必要です。

protocol (読み書き) (カスタムオリジンのみ)

オリジンに接続するとき CloudFront が使用する接続プロトコル。ここには、`http` または `https` が表示されます。

readTimeout (読み書き) (カスタムオリジンのみ)

オリジンにリクエストを送信した後、CloudFront がレスポンスを待機する時間 (秒単位)。これは、レスポンスのパケットを受信してから次のパケットを受信するまで CloudFront が待機する時間も指定します。この値には、4 ~ 60 の範囲の数値を指定する必要があります。

ユースケースに 60 秒を超える時間が必要な場合は、`Response timeout per origin` のクォータの引き上げをリクエストできます。詳細については、「ディストリビューションの一般的なクォータ」を参照してください。

`sslProtocols` (読み書き) (カスタムオリジンのみ)

オリジンとの HTTPS 接続を確立するときに CloudFront が使用できる最小限の SSL/TLS プロトコル。値は、`TLSv1.2`、`TLSv1.1`、`TLSv1`、または `SSLv3` のいずれかです。

`authMethod` (読み取り/書き込み) (Amazon S3 オリジンのみ)

オリジンアクセスアイデンティティ (OAI) を使用している場合は、このフィールドを `origin-access-identity` に設定します。OAI を使用していない場合は、`none` に設定します。`authMethod` を `origin-access-identity` に設定した場合、いくつかの要件があります。

- `region` を指定する必要があります (次のフィールドを参照)。
- リクエストをある Amazon S3 オリジンから別のオリジンに変更する場合は、同じ OAI を使用する必要があります。
- リクエストをカスタムオリジンから Amazon S3 オリジンに変更する場合、OAI を使用することはできません。

> **Note**
>
> このフィールドはオリジンアクセスコントロール (OAC) をサポートしていません。

`region` (読み取り/書き込み) (Amazon S3 オリジンのみ)

Amazon S3 バケットの AWS リージョン。これは、`authMethod` を `origin-access-identity` に設定した場合にのみ必要です。

レスポンスイベント

以下のトピックでは、ビューワーおよびオリジンレスポンスイベントの Lambda 関数に CloudFront が渡すオブジェクトの構造を示します。例に続いて、ビューワーとオリジンレスポンスイベントで使用可能なすべてのフィールドのリストを示します。

トピック

- オリジンレスポンスの例
- ビューワーレスポンスの例
- レスポンスイベントのフィールド

オリジンレスポンスの例

次の例は、オリジンレスポンスイベントオブジェクトを示しています。

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "origin-response",
          "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "x-forwarded-for": [
              {
                "key": "X-Forwarded-For",
                "value": "203.0.113.178"
              }
            ],
            "user-agent": [
              {
                "key": "User-Agent",
                "value": "Amazon CloudFront"
              }
            ],
            "via": [
              {
                "key": "Via",
                "value": "2.0 8f22423015641505b8c857a37450d6c0.cloudfront.net
 (CloudFront)"
              }
            ],
            "host": [
              {
                "key": "Host",
                "value": "example.org"
              }
            ],
            "cache-control": [
              {
```

```
            "key": "Cache-Control",
            "value": "no-cache"
          }
        ]
      },
      "method": "GET",
      "origin": {
        "custom": {
          "customHeaders": {},
          "domainName": "example.org",
          "keepaliveTimeout": 5,
          "path": "",
          "port": 443,
          "protocol": "https",
          "readTimeout": 30,
          "sslProtocols": [
            "TLSv1",
            "TLSv1.1",
            "TLSv1.2"
          ]
        }
      },
      "querystring": "",
      "uri": "/"
    },
    "response": {
      "headers": {
        "access-control-allow-credentials": [
          {
            "key": "Access-Control-Allow-Credentials",
            "value": "true"
          }
        ],
        "access-control-allow-origin": [
          {
            "key": "Access-Control-Allow-Origin",
            "value": "*"
          }
        ],
        "date": [
          {
            "key": "Date",
            "value": "Mon, 13 Jan 2020 20:12:38 GMT"
          }
```

```
            ],
            "referrer-policy": [
              {
                "key": "Referrer-Policy",
                "value": "no-referrer-when-downgrade"
              }
            ],
            "server": [
              {
                "key": "Server",
                "value": "ExampleCustomOriginServer"
              }
            ],
            "x-content-type-options": [
              {
                "key": "X-Content-Type-Options",
                "value": "nosniff"
              }
            ],
            "x-frame-options": [
              {
                "key": "X-Frame-Options",
                "value": "DENY"
              }
            ],
            "x-xss-protection": [
              {
                "key": "X-XSS-Protection",
                "value": "1; mode=block"
              }
            ],
            "content-type": [
              {
                "key": "Content-Type",
                "value": "text/html; charset=utf-8"
              }
            ],
            "content-length": [
              {
                "key": "Content-Length",
                "value": "9593"
              }
            ]
          },
```

```
          "status": "200",
          "statusDescription": "OK"
        }
      }
    }
  ]
}
```

ビューワーレスポンスの例

次の例は、ビューワーレスポンスイベントオブジェクトを示しています。

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "viewer-response",
          "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "host": [
              {
                "key": "Host",
                "value": "d111111abcdef8.cloudfront.net"
              }
            ],
            "user-agent": [
              {
                "key": "User-Agent",
                "value": "curl/7.66.0"
              }
            ],
            "accept": [
              {
                "key": "accept",
                "value": "*/*"
              }
            ]
          },
```

```
          "method": "GET",
          "querystring": "",
          "uri": "/"
        },
        "response": {
          "headers": {
            "access-control-allow-credentials": [
              {
                "key": "Access-Control-Allow-Credentials",
                "value": "true"
              }
            ],
            "access-control-allow-origin": [
              {
                "key": "Access-Control-Allow-Origin",
                "value": "*"
              }
            ],
            "date": [
              {
                "key": "Date",
                "value": "Mon, 13 Jan 2020 20:14:56 GMT"
              }
            ],
            "referrer-policy": [
              {
                "key": "Referrer-Policy",
                "value": "no-referrer-when-downgrade"
              }
            ],
            "server": [
              {
                "key": "Server",
                "value": "ExampleCustomOriginServer"
              }
            ],
            "x-content-type-options": [
              {
                "key": "X-Content-Type-Options",
                "value": "nosniff"
              }
            ],
            "x-frame-options": [
              {
```

```
                "key": "X-Frame-Options",
                "value": "DENY"
              }
            ],
            "x-xss-protection": [
              {
                "key": "X-XSS-Protection",
                "value": "1; mode=block"
              }
            ],
            "age": [
              {
                "key": "Age",
                "value": "2402"
              }
            ],
            "content-type": [
              {
                "key": "Content-Type",
                "value": "text/html; charset=utf-8"
              }
            ],
            "content-length": [
              {
                "key": "Content-Length",
                "value": "9593"
              }
            ]
          },
          "status": "200",
          "statusDescription": "OK"
        }
      }
    }
  ]
}
```

レスポンスイベントのフィールド

レスポンスイベントオブジェクトデータは、config (Records.cf.config)、request (Records.cf.request)、response (Records.cf.response) の 3 つのサブオブジェクトに含まれています。リクエストオブジェクトのフィールドの詳細については、「リクエストオブジェクトの

フィールド」を参照してください。次のリストでは、config および response サブオブジェクトのフィールドについて説明します。

設定オブジェクトのフィールド

次のリストでは、config オブジェクト (Records.cf.config) のフィールドについて説明します。

distributionDomainName (読み取り専用)

レスポンスに関連付けられているディストリビューションのドメイン名。

distributionID (読み取り専用)

レスポンスに関連付けられているディストリビューションの ID。

eventType (読み取り専用)

レスポンスに関連付けられているトリガーのタイプ (origin-response または viewer-response)。

requestId (読み取り専用)

このレスポンスが関連付けられているビューワーから CloudFront へのリクエストを一意に識別する暗号化された文字列。requestId の値は CloudFront アクセスログにも x-edge-request-id として表示されます。詳細については、標準ログ (アクセスログ) の設定および使用および標準ログファイルフィールドを参照してください。

レスポンスオブジェクトのフィールド

次のリストでは、response オブジェクト (Records.cf.response) のフィールドについて説明します。Lambda@Edge 関数を使用して HTTP レスポンスを生成する方法については、「リクエストトリガーでの HTTP レスポンスを生成する」を参照してください。

headers (読み書き)

レスポンスのヘッダー。次の点に注意してください。

- headers オブジェクトのキーは標準の HTTP ヘッダー名を小文字にしたものです。小文字のキーを使用して、大文字と小文字を区別せずにヘッダー値にアクセスできます。
- 各ヘッダーオブジェクト (headers["content-type"]、headers["content-length"] など) はキーと値のペアの配列です。返されたヘッダーの配列には、レスポンスの値ごとに 1 つのキーと値のペアが含まれます。

- key には、HTTP レスポンスに表示される際に大文字と小文字が区別されるヘッダー名が含まれます (Content-Type、Content-Length、Cookie など)。
- value には、HTTP レスポンスに表示されるヘッダー値が含まれます。
- Lambda 関数がレスポンスヘッダーを追加または変更し、ヘッダー key フィールドを含めない場合、Lambda@Edge は指定したヘッダー名を使用してヘッダー key を自動的に挿入します。ヘッダー名をどのようにフォーマットしたかにかかわらず、自動的に挿入されるヘッダーキーの各パートは、先頭が大文字になり、ハイフン (-) で区切られます。

  たとえば、ヘッダー key なしで次のようなヘッダーを追加できます。

```
"content-type": [
  {
    "value": "text/html;charset=UTF-8"
  }
]
```

  この例では、Lambda@Edge は "key": "Content-Type" を自動的に挿入します。

ヘッダー使用の制限の詳細については、「エッジ関数に対する制限」を参照してください。

status

レスポンスの HTTP ステータスコード。

statusDescription

レスポンスの HTTP ステータスの説明。

リクエストとレスポンスを使用する

このセクションのトピックでは、Lambda@Edge リクエストとレスポンスを使用するいくつかの方法について説明します。

トピック

- オリジンフェイルオーバーで Lambda@Edge 関数を使用する
- リクエストトリガーでの HTTP レスポンスを生成する
- オリジンレスポンストリガーでの HTTP レスポンスを更新する
- include body オプションを選択してリクエストボディにアクセスする

オリジンフェイルオーバーで Lambda@Edge 関数を使用する

例えば、高い可用性を確保するために設定したオリジンフェイルオーバーなど、オリジングループで設定した CloudFront ディストリビューションで Lambda@Edge 関数を使用できます。オリジングループで Lambda 関数を使用するには、キャッシュ動作を作成するときにオリジングループのオリジンリクエストまたはオリジンレスポンストリガーで関数を指定します。

詳細については、次を参照してください:

- オリジングループを作成する: オリジングループを作成する
- Lambda@Edge でのオリジンフェイルオーバーの機能 Lambda@Edge 関数でのオリジンフェイルオーバーの使用

リクエストトリガーでの HTTP レスポンスを生成する

CloudFront がリクエストを受け取ったときに、Lambda 関数が HTTP レスポンスを生成することで、CloudFront がレスポンスをオリジンに転送せずに直接ビューワーに返すようにできます。HTTP レスポンスを生成することで、オリジンの負荷が軽減され、通常はビューワーのレイテンシーも短縮されます。

以下に示しているのは、HTTP レスポンスを生成する一般的なシナリオです。

- 小さいウェブページをビューワーに返す。
- HTTP 301 または 302 ステータスコードを返して、ユーザーを別のウェブページにリダイレクトする。
- ユーザーが認証されない場合に HTTP 401 ステータスコードをビューワーに返す。

Lambda@Edge 関数は、以下の CloudFront イベントが発生したときに HTTP レスポンスを生成できます。

ビューワーリクエストイベント

関数がビューワーリクエストイベントによってトリガーされると、CloudFront はレスポンスをビューワーに返し、キャッシュしません。

オリジンリクエストイベント

関数がオリジンリクエストイベントによってトリガーされると、CloudFront は、その関数によって以前に生成されたレスポンスがエッジキャッシュ内にあるかどうかを確認します。

- レスポンスがキャッシュ内にある場合、関数は実行されず、CloudFront はキャッシュされたレスポンスをビューワーに返します。
- レスポンスがキャッシュ内にない場合、関数が実行され、CloudFront はそのレスポンスをビューワーに返すとともに、キャッシュします。

HTTP レスポンスを生成するためのサンプルコードを見るには、「Lambda@Edge 関数の例」を参照してください。レスポンストリガーの HTTP レスポンスを置き換えることもできます。詳細については、「オリジンレスポンストリガーでの HTTP レスポンスを更新する」を参照してください。

プログラミングモデル

このセクションでは、Lambda@Edge を使用して HTTP レスポンスを生成するためのプログラミングモデルについて説明します。

トピック

- レスポンスオブジェクト
- エラー
- 必須フィールド

レスポンスオブジェクト

`result` メソッドの `callback` パラメータとして返すレスポンスには、以下の構造が必要です (`status` フィールドのみが必須)。

```
const response = {
    body: 'content',
    bodyEncoding: 'text' | 'base64',
    headers: {
        'header name in lowercase': [{
            key: 'header name in standard case',
            value: 'header value'
         }],
         ...
    },
    status: 'HTTP status code (string)',
    statusDescription: 'status description'
};
```

レスポンスオブジェクトには、以下の値が含まれる場合があります。

body

生成されたレスポンスで CloudFront が返す本文 (存在する場合)。

bodyEncoding

body で指定した値のエンコード。有効なエンコードは `text` と `base64` のみです。`response` オブジェクトに body を含めるが、`bodyEncoding` を省略した場合、CloudFront は本文をテキストとして扱います。

`bodyEncoding` を `base64` と指定したが本文が有効な base64 でない場合、CloudFront はエラーを返します。

headers

生成されるレスポンスで CloudFront が返すヘッダー。次の点に注意してください。

- `headers` オブジェクトのキーは標準の HTTP ヘッダー名を小文字にしたものです。小文字のキーを使用して、大文字と小文字を区別せずにヘッダー値にアクセスできます。
- 各ヘッダー (`headers["accept"]`、`headers["host"]` など) はキーと値のペアの配列です。返されたヘッダーの配列には、生成されたレスポンスの値ごとに 1 つのキーと値のペアが含まれます。
- key (省略可能) は、HTTP リクエストに表示されるヘッダーの大文字と小文字を区別する名前です (`accept`、`host` など)。
- ヘッダー値として `value` を指定します。
- キーと値のペアのヘッダーキー部分を含めない場合、Lambda@Edge は指定したヘッダー名を使用してヘッダーキーを自動的に挿入します。ヘッダー名をどのようにフォーマットしたかにかかわらず、挿入されるヘッダーキーは、各パートの先頭の大文字がハイフン (-) で区切られて自動的にフォーマットされます。

  たとえば、ヘッダーキー `'content-type': [{ value: 'text/html;charset=UTF-8' }]` なしで次のようなヘッダーを追加できます。

  この例で、Lambda@Edge はヘッダーキー `Content-Type` を作成します。

ヘッダー使用の制限の詳細については、「エッジ関数に対する制限」を参照してください。

status

HTTP ステータスコード 。ステータスコードを文字列として指定します。CloudFront は、提供されているステータスコードを、以下に使用します。

- レスポンスでの返却

- オリジンリクエストイベントによってトリガーされた関数によってレスポンスが生成されたときの、CloudFront エッジキャッシュへの保存
- CloudFront へのログイン 標準ログ (アクセスログ) の設定および使用

`status` の値が 200 ~ 599 の範囲にない場合、CloudFront はエラーをビューワーに返します。

`statusDescription`

CloudFront がレスポンスで HTTP ステータスコードに付けて返す説明。標準の説明 (HTTP ステータスコード 200 の場合の OK など) を使用する必要はありません。

エラー

生成された HTTP レスポンスで発生する可能性があるエラーを以下に示します。

レスポンスに本文が含まれ、ステータスに 204 (No Content) が指定されている

関数がビューワーリクエストによってトリガーされると、CloudFront は、以下の両方が当てはまる場合に、HTTP 502 ステータスコード (Bad Gateway) をビューワーに返します。

- `status` の値が 204 (No Content) である。
- このレスポンスに `body` の値が含まれている。

これは、`HTTP 204` レスポンスにはメッセージ本文を含める必要がないことを述べている RFC 2616 のオプションの制限を Lambda@Edge が適用しているためです。

生成されるレスポンスのサイズ制限を超えている

Lambda 関数によって生成されるレスポンスの最大サイズは、関数をトリガーするイベントによって異なります。

- ビューワーリクエストイベント - 40 KB
- オリジンリクエストイベント - 1 MB

レスポンスがこの許容サイズを超えると、CloudFront が HTTP 502 ステータスコード (Bad Gateway) をビューワーに返します。

必須フィールド

`status` フィールドは必須です。

その他のすべてのフィールドはオプションです。

オリジンレスポンストリガーでの HTTP レスポンスを更新する

CloudFront がオリジンサーバーから HTTP レスポンスを受け取ったときに、キャッシュ動作に関連付けられた origin-response トリガーがあれば、HTTP レスポンスを変更して、オリジンから返されたものを上書きできます。

以下に示しているのは、HTTP レスポンスを更新する一般的なシナリオです。

- オリジンがエラーステータスコード (4xx または 5xx) を返すと、ステータスを変更して HTTP 200 ステータスコードを設定し、ビューワーに返す静的な本文コンテンツを作成する。サンプルコードについては、「例: オリジンレスポンストリガーを使用してエラーステータスコードを 200 に更新する」を参照してください。
- オリジンがエラーステータスコード (4xx または 5xx) を返すと、ステータスを変更して HTTP 301 または HTTP 302 ステータスコードを設定し、ユーザーを別のウェブサイトにリダイレクトする。サンプルコードについては、「例: オリジンレスポンストリガーを使用してエラーステータスコードを 302 に更新する」を参照してください。

> **Note**
>
> 関数は `200` と `599` の間 (両端を含む) のステータス値を返す必要があります。そうでない場合、CloudFront はエラーをビューワーに返します。

ビューワーとオリジンのリクエストイベントの HTTP レスポンスを置き換えることもできます。詳細については、「リクエストトリガーでの HTTP レスポンスを生成する」を参照してください。

HTTP レスポンスを使用する場合、Lambda@Edge は、オリジンサーバーから返された本文を origin-response トリガーに公開しません。必要な値に設定することで静的なコンテンツ本文を生成したり、値を空に設定することで関数内の本文を削除したりできます。関数内の本文フィールドを更新しない場合は、オリジンサーバーによって返された元の本文がビューワーに返されます。

include body オプションを選択してリクエストボディにアクセスする

書き込み可能な HTTP メソッド (POST、PUT、DELETE など) のリクエストのボディを Lambda@Edge で公開することを選択できるため、Lambda 関数でそのボディにアクセスできます。読み取り専用アクセスを選択することも、ボディを置き換えることを指定することもできます。

このオプションを有効にするには、ビューワーリクエストやオリジンリクエストイベントのために関数に CloudFront トリガーを作成するとき、[ボディを含める] を選択します。詳細については、

「Lambda@Edge 関数のトリガーを追加する」を参照してください。または、関数で [ボディを含める] を使用する方法については、「Lambda@Edge イベント構造」を参照してください。

この機能を使用する場合のシナリオには次のようなものがあります。

- お客様の入力データをオリジンサーバーに返送することなく、「お問い合わせ」フォームのようなウェブフォームを処理します。
- ビューワーブラウザによって送信されるウェブビーコンデータを収集し、エッジで処理します。

サンプルコードについては、「Lambda@Edge 関数の例」を参照してください。

> **Note**
>
> リクエストボディが大きい場合は、Lambda@Edge によって切り捨てられます。最大サイズ制限と切り捨ての詳細については、「Include Body オプションがあるリクエストボディに対する制限」を参照してください。

Lambda@Edge 関数の例

Amazon CloudFront での Lambda 関数の使用例については、以下のセクションを参照してください。

> **Note**
>
> Lambda@Edge 関数にランタイム Node.js 18 以降を選択すると、`index.mjs` ファイルが自動的に作成されます。次のコード例を使用するには、`index.mjs` ファイルの名前を `index.js` に変更します。

トピック

- 一般的な例
- レスポンスを生成する - 例
- クエリ文字列 - 例
- 国またはデバイスタイプヘッダー別のコンテンツのパーソナライズ - 例
- コンテンツベースの動的オリジンの選択 - 例
- エラーステータスを更新する - 例

- リクエストボディにアクセスする - 例

一般的な例

このセクションの例では、CloudFront で Lambda@Edge を使用する一般的な方法をいくつか示しています。

トピック

- 例: A/B テスト
- 例: レスポンスヘッダーをオーバーライドする

例: A/B テスト

次の例を使用すると、リダイレクトを作成したり URL を変更したりすることなく、2 つの異なるバージョンのイメージをテストできます。この例では、ビューワーリクエスト内の Cookie を読み取り、それに応じてリクエスト URL を変更します。ビューワーがいずれかの期待値を使用して Cookie を送信しない場合、例ではビューワーを URL のいずれかにランダムに割り当てます。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const headers = request.headers;

    if (request.uri !== '/experiment-pixel.jpg') {
        // do not process if this is not an A-B test request
        callback(null, request);
        return;
    }

    const cookieExperimentA = 'X-Experiment-Name=A';
    const cookieExperimentB = 'X-Experiment-Name=B';
    const pathExperimentA = '/experiment-group/control-pixel.jpg';
    const pathExperimentB = '/experiment-group/treatment-pixel.jpg';

    /*
     * Lambda at the Edge headers are array objects.
     *
```

```
     * Client may send multiple Cookie headers, i.e.:
     * > GET /viewerRes/test HTTP/1.1
     * > User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1
 OpenSSL/1.0.1u zlib/1.2.3
     * > Cookie: First=1; Second=2
     * > Cookie: ClientCode=abc
     * > Host: example.com
     *
     * You can access the first Cookie header at headers["cookie"][0].value
     * and the second at headers["cookie"][1].value.
     *
     * Header values are not parsed. In the example above,
     * headers["cookie"][0].value is equal to "First=1; Second=2"
     */
    let experimentUri;
    if (headers.cookie) {
        for (let i = 0; i < headers.cookie.length; i++) {
            if (headers.cookie[i].value.indexOf(cookieExperimentA) >= 0) {
                console.log('Experiment A cookie found');
                experimentUri = pathExperimentA;
                break;
            } else if (headers.cookie[i].value.indexOf(cookieExperimentB) >= 0) {
                console.log('Experiment B cookie found');
                experimentUri = pathExperimentB;
                break;
            }
        }
    }

    if (!experimentUri) {
        console.log('Experiment cookie has not been found. Throwing dice...');
        if (Math.random() < 0.75) {
            experimentUri = pathExperimentA;
        } else {
            experimentUri = pathExperimentB;
        }
    }

    request.uri = experimentUri;
    console.log(`Request uri set to "${request.uri}"`);
    callback(null, request);
};
```

Python

```
import json
import random

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    if request['uri'] != '/experiment-pixel.jpg':
        # Not an A/B Test
        return request

    cookieExperimentA, cookieExperimentB = 'X-Experiment-Name=A', 'X-Experiment-
Name=B'
    pathExperimentA, pathExperimentB = '/experiment-group/control-pixel.jpg', '/
experiment-group/treatment-pixel.jpg'

    '''
    Lambda at the Edge headers are array objects.

    Client may send multiple cookie headers. For example:
    > GET /viewerRes/test HTTP/1.1
    > User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1
 OpenSSL/1.0.1u zlib/1.2.3
    > Cookie: First=1; Second=2
    > Cookie: ClientCode=abc
    > Host: example.com

    You can access the first Cookie header at headers["cookie"][0].value
    and the second at headers["cookie"][1].value.

    Header values are not parsed. In the example above,
    headers["cookie"][0].value is equal to "First=1; Second=2"
    '''

    experimentUri = ""

    for cookie in headers.get('cookie', []):
        if cookieExperimentA in cookie['value']:
            print("Experiment A cookie found")
            experimentUri = pathExperimentA
            break
        elif cookieExperimentB in cookie['value']:
```

```
            print("Experiment B cookie found")
            experimentUri = pathExperimentB
            break

    if not experimentUri:
        print("Experiment cookie has not been found. Throwing dice...")
        if random.random() < 0.75:
            experimentUri = pathExperimentA
        else:
            experimentUri = pathExperimentB

    request['uri'] = experimentUri
    print(f"Request uri set to {experimentUri}")
    return request
```

例: レスポンスヘッダーをオーバーライドする

以下の例は、レスポンスヘッダーの値を別のヘッダーの値に基づいて変更する方法を示しています。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
   const response = event.Records[0].cf.response;
   const headers = response.headers;

   const headerNameSrc = 'X-Amz-Meta-Last-Modified';
   const headerNameDst = 'Last-Modified';

   if (headers[headerNameSrc.toLowerCase()]) {
      headers[headerNameDst.toLowerCase()] = [
         headers[headerNameSrc.toLowerCase()][0],
      ];
      console.log(`Response header "${headerNameDst}" was set to ` +
               `"${headers[headerNameDst.toLowerCase()][0].value}"`);
   }

   callback(null, response);
};
```

Python

```
import json

def lambda_handler(event, context):
    response = event["Records"][0]["cf"]["response"]
    headers = response["headers"]

    headerNameSrc = "X-Amz-Meta-Last-Modified"
    headerNameDst = "Last-Modified"

    if headers.get(headerNameSrc.lower(), None):
        headers[headerNameDst.lower()] = [headers[headerNameSrc.lower()][0]]
        print(f"Response header {headerNameDst.lower()} was set to
 {headers[headerNameSrc.lower()][0]}")

    return response
```

レスポンスを生成する - 例

このセクションの例では、Lambda@Edge を使用してレスポンスを生成する方法を示しています。

トピック

- 例: 静的コンテンツを提供する (生成されたレスポンス)
- 例: HTTP リダイレクトを生成する (生成されたレスポンス)

例: 静的コンテンツを提供する (生成されたレスポンス)

次の例は、Lambda 関数を使用して静的ウェブサイトコンテンツを提供する方法を示しています。これにより、オリジンサーバーの負荷と全体的なレイテンシーが軽減されます。

> **Note**
>
> HTTP レスポンスは、ビューワーリクエストおよびオリジンリクエストのイベントに対して生成できます。詳細については、「the section called “リクエストトリガーでの HTTP レスポンスを生成する”」を参照してください。

オリジンレスポンスイベントで HTTP レスポンスのボディを置き換えたり、削除することもできます。詳細については、「the section called “オリジンレスポンストリガーでの HTTP レスポンスを更新する”」を参照してください。

Node.js

```
'use strict';

const content = `
<\!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Simple Lambda@Edge Static Content Response</title>
  </head>
  <body>
    <p>Hello from Lambda@Edge!</p>
  </body>
</html>
`;

exports.handler = (event, context, callback) => {
    /*
     * Generate HTTP OK response using 200 status code with HTML body.
     */
    const response = {
        status: '200',
        statusDescription: 'OK',
        headers: {
            'cache-control': [{
                key: 'Cache-Control',
                value: 'max-age=100'
            }],
            'content-type': [{
                key: 'Content-Type',
                value: 'text/html'
            }]
        },
        body: content,
    };
    callback(null, response);
```

```
};
```

Python

```
import json

CONTENT = """
<\!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Simple Lambda@Edge Static Content Response</title>
</head>
<body>
    <p>Hello from Lambda@Edge!</p>
</body>
</html>
"""

def lambda_handler(event, context):
    # Generate HTTP OK response using 200 status code with HTML body.
    response = {
        'status': '200',
        'statusDescription': 'OK',
        'headers': {
            'cache-control': [
                {
                    'key': 'Cache-Control',
                    'value': 'max-age=100'
                }
            ],
            "content-type": [
                {
                    'key': 'Content-Type',
                    'value': 'text/html'
                }
            ]
        },
        'body': CONTENT
    }
    return response
```

例: HTTP リダイレクトを生成する (生成されたレスポンス)

次の例は、HTTP リダイレクトを生成する方法を示しています。

> **Note**
>
> HTTP レスポンスは、ビューワーリクエストおよびオリジンリクエストのイベントに対して生成できます。詳細については、「リクエストトリガーでの HTTP レスポンスを生成する」を参照してください。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    /*
     * Generate HTTP redirect response with 302 status code and Location header.
     */
    const response = {
        status: '302',
        statusDescription: 'Found',
        headers: {
            location: [{
                key: 'Location',
                value: 'https://docs.aws.amazon.com/lambda/latest/dg/lambda-
edge.html',
            }],
        },
    };
    callback(null, response);
};
```

Python

```
def lambda_handler(event, context):

    # Generate HTTP redirect response with 302 status code and Location header.

    response = {
        'status': '302',
        'statusDescription': 'Found',
```

```
        'headers': {
            'location': [{
                'key': 'Location',
                'value': 'https://docs.aws.amazon.com/lambda/latest/dg/lambda-
edge.html'
            }]
        }
    }

    return response
```

クエリ文字列 - 例

このセクションの例には、クエリ文字列で Lambda@Edge を使用する方法が含まれています。

トピック

- 例: クエリ文字列パラメータに基づくヘッダーを追加する
- 例: キャッシュヒット率を向上させるためにクエリ文字列パラメータを正規化する
- 例: 認証されていないユーザーをサインインページにリダイレクトする

例: クエリ文字列パラメータに基づくヘッダーを追加する

以下の例では、クエリ文字列パラメータのキーと値のペアを取得してから、それらの値に基づいてヘッダーを追加する方法を示します。

Node.js

```
'use strict';

const querystring = require('querystring');
exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;

    /* When a request contains a query string key-value pair but the origin server
     * expects the value in a header, you can use this Lambda function to
     * convert the key-value pair to a header. Here's what the function does:
     * 1. Parses the query string and gets the key-value pair.
     * 2. Adds a header to the request using the key-value pair that the function
 got in step 1.
     */
```

```
    /* Parse request querystring to get javascript object */
    const params = querystring.parse(request.querystring);

    /* Move auth param from querystring to headers */
    const headerName = 'Auth-Header';
    request.headers[headerName.toLowerCase()] = [{ key: headerName, value:
 params.auth }];
    delete params.auth;

    /* Update request querystring */
    request.querystring = querystring.stringify(params);

    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs, urlencode

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    '''
    When a request contains a query string key-value pair but the origin server
    expects the value in a header, you can use this Lambda function to
    convert the key-value pair to a header. Here's what the function does:
        1. Parses the query string and gets the key-value pair.
        2. Adds a header to the request using the key-value pair that the function
 got in step 1.
    '''

    # Parse request querystring to get dictionary/json
    params = {k : v[0] for k, v in parse_qs(request['querystring']).items()}

    # Move auth param from querystring to headers
    headerName = 'Auth-Header'
    request['headers'][headerName.lower()] = [{'key': headerName, 'value':
 params['auth']}]
    del params['auth']

    # Update request querystring
    request['querystring'] = urlencode(params)
```

```
    return request
```

例: キャッシュヒット率を向上させるためにクエリ文字列パラメータを正規化する

次の例では、CloudFront がリクエストをオリジンに転送する前にクエリ文字列に以下の変更を行うことで、キャッシュヒット率を向上させる方法を示します。

- パラメータの名前によりキーと値のペアをアルファベット順に並べ替える
- キーと値のペアを小文字に変更する

詳細については、「クエリ文字列パラメータに基づいてコンテンツをキャッシュする」を参照してください。

Node.js

```
'use strict';

const querystring = require('querystring');

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    /* When you configure a distribution to forward query strings to the origin and
     * to cache based on an allowlist of query string parameters, we recommend
     * the following to improve the cache-hit ratio:
     * - Always list parameters in the same order.
     * - Use the same case for parameter names and values.
     *
     * This function normalizes query strings so that parameter names and values
     * are lowercase and parameter names are in alphabetical order.
     *
     * For more information, see:
     * https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/
QueryStringParameters.html
     */

    console.log('Query String: ', request.querystring);

    /* Parse request query string to get javascript object */
    const params = querystring.parse(request.querystring.toLowerCase());
    const sortedParams = {};
```

```
    /* Sort param keys */
    Object.keys(params).sort().forEach(key => {
        sortedParams[key] = params[key];
    });

    /* Update request querystring with normalized  */
    request.querystring = querystring.stringify(sortedParams);

    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs, urlencode

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    '''
    When you configure a distribution to forward query strings to the origin and
    to cache based on an allowlist of query string parameters, we recommend
    the following to improve the cache-hit ratio:
    Always list parameters in the same order.
    - Use the same case for parameter names and values.

    This function normalizes query strings so that parameter names and values
    are lowercase and parameter names are in alphabetical order.

    For more information, see:
    https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/
QueryStringParameters.html
    '''
    print("Query string: ", request["querystring"])

    # Parse request query string to get js object
    params = {k : v[0] for k, v in parse_qs(request['querystring'].lower()).items()}

    # Sort param keys
    sortedParams = sorted(params.items(), key=lambda x: x[0])

    # Update request querystring with normalized
    request['querystring'] = urlencode(sortedParams)
```

```
    return request
```

例: 認証されていないユーザーをサインインページにリダイレクトする

次の例では、ユーザーが認証情報を入力していない場合にサインインページにリダイレクトする方法を示します。

Node.js

```
'use strict';

function parseCookies(headers) {
    const parsedCookie = {};
    if (headers.cookie) {
        headers.cookie[0].value.split(';').forEach((cookie) => {
            if (cookie) {
                const parts = cookie.split('=');
                parsedCookie[parts[0].trim()] = parts[1].trim();
            }
        });
    }
    return parsedCookie;
}

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const headers = request.headers;

    /* Check for session-id in request cookie in viewer-request event,
     * if session-id is absent, redirect the user to sign in page with original
     * request sent as redirect_url in query params.
     */

    /* Check for session-id in cookie, if present then proceed with request */
    const parsedCookies = parseCookies(headers);
    if (parsedCookies && parsedCookies['session-id']) {
        callback(null, request);
        return;
    }

    /* URI encode the original request to be sent as redirect_url in query params */
```

```
    const encodedRedirectUrl = encodeURIComponent(`https://
${headers.host[0].value}${request.uri}?${request.querystring}`);
    const response = {
        status: '302',
        statusDescription: 'Found',
        headers: {
            location: [{
                key: 'Location',
                value: `https://www.example.com/signin?redirect_url=
${encodedRedirectUrl}`,
            }],
        },
    };
    callback(null, response);
};
```

Python

```
import urllib

def parseCookies(headers):
    parsedCookie = {}
    if headers.get('cookie'):
        for cookie in headers['cookie'][0]['value'].split(';'):
            if cookie:
                parts = cookie.split('=')
                parsedCookie[parts[0].strip()] = parts[1].strip()
    return parsedCookie

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    '''
    Check for session-id in request cookie in viewer-request event,
    if session-id is absent, redirect the user to sign in page with original
    request sent as redirect_url in query params.
    '''

    # Check for session-id in cookie, if present, then proceed with request
    parsedCookies = parseCookies(headers)

    if parsedCookies and parsedCookies['session-id']:
```

```
        return request

    # URI encode the original request to be sent as redirect_url in query params
    redirectUrl = "https://%s%s?%s" % (headers['host'][0]['value'], request['uri'],
request['querystring'])
    encodedRedirectUrl = urllib.parse.quote_plus(redirectUrl.encode('utf-8'))

    response = {
        'status': '302',
        'statusDescription': 'Found',
        'headers': {
            'location': [{
                'key': 'Location',
                'value': 'https://www.example.com/signin?redirect_url=%s' %
encodedRedirectUrl
            }]
        }
    }
    return response
```

国またはデバイスタイプヘッダー別のコンテンツのパーソナライズ - 例

このセクションの例では、Lambda@Edge を使用し、ビューワーが使用しているデバイスの場所またはタイプに基づいて動作をカスタマイズする方法を示しています。

トピック

- 例: ビューワーリクエストを国に固有の URL にリダイレクトする
- 例: デバイスに基づいて異なるバージョンのオブジェクトを供給する

例: ビューワーリクエストを国に固有の URL にリダイレクトする

次の例では、HTTP リダイレクト応答を国に固有の URL で生成し、ビューワーにレスポンスを返す方法を示します。これは、国ごとに異なる応答を提供する場合に便利です。次に例を示します。

- 国別のサブドメイン (us.example.com および tw.example.com など) がある場合は、ビューワーが example.com をリクエストしたときにリダイレクト応答を生成できます。
- 動画をストリーミングしていて、そのコンテンツを特定の国でストリーミングする権限がない場合は、その国のユーザーを別のページにリダイレクトして動画を閲覧できない理由について説明できます。

次の点に注意してください。

- CloudFront-Viewer-Country ヘッダーに基づいてキャッシュするようにディストリビューションを設定する必要があります。詳細については、「選択されたリクエストヘッダーに基づいたキャッシュ」を参照してください。
- CloudFront は、ビューワーリクエストイベントの後に CloudFront-Viewer-Country ヘッダーを追加します。この例を使用するには、オリジンリクエストイベントのトリガーを作成する必要があります。

Node.js

```
'use strict';

/* This is an origin request function */
exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const headers = request.headers;

    /*
     * Based on the value of the CloudFront-Viewer-Country header, generate an
     * HTTP status code 302 (Redirect) response, and return a country-specific
     * URL in the Location header.
     * NOTE: 1. You must configure your distribution to cache based on the
     *          CloudFront-Viewer-Country header. For more information, see
     *          https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-
headers
     *       2. CloudFront adds the CloudFront-Viewer-Country header after the
 viewer
     *          request event. To use this example, you must create a trigger for
 the
     *          origin request event.
     */

    let url = 'https://example.com/';
    if (headers['cloudfront-viewer-country']) {
        const countryCode = headers['cloudfront-viewer-country'][0].value;
        if (countryCode === 'TW') {
            url = 'https://tw.example.com/';
        } else if (countryCode === 'US') {
            url = 'https://us.example.com/';
        }
    }
```

```
    const response = {
        status: '302',
        statusDescription: 'Found',
        headers: {
            location: [{
                key: 'Location',
                value: url,
            }],
        },
    };
    callback(null, response);
};
```

Python

```
# This is an origin request function

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    '''
    Based on the value of the CloudFront-Viewer-Country header, generate an
    HTTP status code 302 (Redirect) response, and return a country-specific
    URL in the Location header.
    NOTE: 1. You must configure your distribution to cache based on the
            CloudFront-Viewer-Country header. For more information, see
            https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
          2. CloudFront adds the CloudFront-Viewer-Country header after the viewer
            request event. To use this example, you must create a trigger for the
            origin request event.
    '''

    url = 'https://example.com/'
    viewerCountry = headers.get('cloudfront-viewer-country')
    if viewerCountry:
        countryCode = viewerCountry[0]['value']
        if countryCode == 'TW':
            url = 'https://tw.example.com/'
        elif countryCode == 'US':
            url = 'https://us.example.com/'
```

```
    response = {
        'status': '302',
        'statusDescription': 'Found',
        'headers': {
            'location': [{
                'key': 'Location',
                'value': url
            }]
        }
    }

    return response
```

例: デバイスに基づいて異なるバージョンのオブジェクトを供給する

次の例では、ユーザーが使用しているモバイルデバイスまたはタブレットのようなデバイスのタイプに基づいてオブジェクトの異なるバージョンを提供する方法を示します。次の点に注意してください。

- CloudFront-Is-*-Viewer ヘッダーに基づいてキャッシュするようにディストリビューションを設定する必要があります。詳細については、「選択されたリクエストヘッダーに基づいたキャッシュ」を参照してください。
- CloudFront は、ビューワーリクエストイベントの後に CloudFront-Is-*-Viewer ヘッダーを追加します。この例を使用するには、オリジンリクエストイベントのトリガーを作成する必要があります。

Node.js

```
'use strict';

/* This is an origin request function */
exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const headers = request.headers;

    /*
     * Serve different versions of an object based on the device type.
     * NOTE: 1. You must configure your distribution to cache based on the
     *          CloudFront-Is-*-Viewer headers. For more information, see
     *          the following documentation:
```

```
     *          https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-
headers
     *          https://docs.aws.amazon.com/console/cloudfront/cache-on-device-type
     *       2. CloudFront adds the CloudFront-Is-*-Viewer headers after the viewer
     *          request event. To use this example, you must create a trigger for
 the
     *          origin request event.
     */

    const desktopPath = '/desktop';
    const mobilePath = '/mobile';
    const tabletPath = '/tablet';
    const smarttvPath = '/smarttv';

    if (headers['cloudfront-is-desktop-viewer']
        && headers['cloudfront-is-desktop-viewer'][0].value === 'true') {
        request.uri = desktopPath + request.uri;
    } else if (headers['cloudfront-is-mobile-viewer']
               && headers['cloudfront-is-mobile-viewer'][0].value === 'true') {
        request.uri = mobilePath + request.uri;
    } else if (headers['cloudfront-is-tablet-viewer']
               && headers['cloudfront-is-tablet-viewer'][0].value === 'true') {
        request.uri = tabletPath + request.uri;
    } else if (headers['cloudfront-is-smarttv-viewer']
               && headers['cloudfront-is-smarttv-viewer'][0].value === 'true') {
        request.uri = smarttvPath + request.uri;
    }
    console.log(`Request uri set to "${request.uri}"`);

    callback(null, request);
};
```

Python

```
# This is an origin request function
def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    '''
    Serve different versions of an object based on the device type.
    NOTE: 1. You must configure your distribution to cache based on the
            CloudFront-Is-*-Viewer headers. For more information, see
```

```
            the following documentation:
            https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
            https://docs.aws.amazon.com/console/cloudfront/cache-on-device-type
          2. CloudFront adds the CloudFront-Is-*-Viewer headers after the viewer
            request event. To use this example, you must create a trigger for the
            origin request event.
    '''

    desktopPath = '/desktop';
    mobilePath = '/mobile';
    tabletPath = '/tablet';
    smarttvPath = '/smarttv';

    if 'cloudfront-is-desktop-viewer' in headers and headers['cloudfront-is-desktop-
viewer'][0]['value'] == 'true':
        request['uri'] = desktopPath + request['uri']
    elif 'cloudfront-is-mobile-viewer' in headers and headers['cloudfront-is-mobile-
viewer'][0]['value'] == 'true':
        request['uri'] = mobilePath + request['uri']
    elif 'cloudfront-is-tablet-viewer' in headers and headers['cloudfront-is-tablet-
viewer'][0]['value'] == 'true':
        request['uri'] = tabletPath + request['uri']
    elif 'cloudfront-is-smarttv-viewer' in headers and headers['cloudfront-is-
smarttv-viewer'][0]['value'] == 'true':
        request['uri'] = smarttvPath + request['uri']

    print("Request uri set to %s" % request['uri'])

    return request
```

コンテンツベースの動的オリジンの選択 - 例

このセクションの例では、Lambda@Edge を使用し、リクエスト内の情報に基づいて異なるオリジンにルーティングする方法を示しています。

トピック

- 例: オリジンリクエストトリガーを使用してカスタムオリジンを Amazon S3 オリジンに変更する
- 例: オリジンリクエストトリガーを使用して Amazon S3 オリジンのリージョンを変更する
- 例: オリジンリクエストトリガーを使用して Amazon S3 オリジンからカスタムオリジンに変更する

- 例: オリジンリクエストトリガーを使用して Amazon S3 バケットから別のバケットにトラフィックを徐々に転送する
- 例: オリジンリクエストトリガーを使用して Country ヘッダーに基づいてオリジンのドメイン名を変更する

例: オリジンリクエストトリガーを使用してカスタムオリジンを Amazon S3 オリジンに変更する

この関数では、origin-request トリガーを使用して、リクエストのプロパティに基づいて、カスタムオリジンから、コンテンツがフェッチされる Amazon S3 オリジンに変更する方法を示しています。

Node.js

```
'use strict';

 const querystring = require('querystring');

 exports.handler = (event, context, callback) => {
     const request = event.Records[0].cf.request;

     /**
      * Reads query string to check if S3 origin should be used, and
      * if true, sets S3 origin properties.
      */

     const params = querystring.parse(request.querystring);

     if (params['useS3Origin']) {
         if (params['useS3Origin'] === 'true') {
             const s3DomainName = 'my-bucket.s3.amazonaws.com';

             /* Set S3 origin fields */
             request.origin = {
                 s3: {
                     domainName: s3DomainName,
                     region: '',
                     authMethod: 'none',
                     path: '',
                     customHeaders: {}
                 }
             };
             request.headers['host'] = [{ key: 'host', value: s3DomainName}];
         }
```

```
     }

    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    '''
    Reads query string to check if S3 origin should be used, and
    if true, sets S3 origin properties
    '''
    params = {k: v[0] for k, v in parse_qs(request['querystring']).items()}
    if params.get('useS3Origin') == 'true':
        s3DomainName = 'my-bucket.s3.amazonaws.com'

        # Set S3 origin fields
        request['origin'] = {
            's3': {
                'domainName': s3DomainName,
                'region': '',
                'authMethod': 'none',
                'path': '',
                'customHeaders': {}
            }
        }
        request['headers']['host'] = [{'key': 'host', 'value': s3DomainName}]
    return request
```

例: オリジンリクエストトリガーを使用して Amazon S3 オリジンのリージョンを変更する

この関数では、origin-request トリガーを使用して、リクエストのプロパティに基づいて、コンテンツがフェッチされる Amazon S3 オリジンを変更する方法を示しています。

この例では、CloudFront-Viewer-Country ヘッダーの値を使用して、S3 バケットのドメイン名を、ビューワーに近いリージョンのバケットに更新します。これは、以下のように役立ちます。

- 指定したリージョンがビューワーの国に近いほど、レイテンシーが短縮されます。

- リクエスト元と同じ国にあるオリジンからデータが提供されることになり、データ主権が確保されます。

この例を使用するには、以下を実行する必要があります。

- CloudFront-Viewer-Country ヘッダーに基づいてキャッシュするようにディストリビューションを設定します。詳細については、「選択されたリクエストヘッダーに基づいたキャッシュ」を参照してください。
- オリジンリクエストイベントでこの関数のトリガーを作成します。CloudFront はビューワーリクエストイベントの後に CloudFront-Viewer-Country ヘッダーを追加するため、この例を使用するには、オリジンリクエストに対して関数が実行されることを確認する必要があります。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;

    /**
     * This blueprint demonstrates how an origin-request trigger can be used to
     * change the origin from which the content is fetched, based on request
 properties.
     * In this example, we use the value of the CloudFront-Viewer-Country header
     * to update the S3 bucket domain name to a bucket in a Region that is closer to
     * the viewer.
     *
     * This can be useful in several ways:
     *      1) Reduces latencies when the Region specified is nearer to the viewer's
     *         country.
     *      2) Provides data sovereignty by making sure that data is served from an
     *         origin that's in the same country that the request came from.
     *
     * NOTE: 1. You must configure your distribution to cache based on the
     *          CloudFront-Viewer-Country header. For more information, see
     *          https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-
headers
     *       2. CloudFront adds the CloudFront-Viewer-Country header after the
 viewer
     *          request event. To use this example, you must create a trigger for
 the
```

```
     *          origin request event.
     */

    const countryToRegion = {
        'DE': 'eu-central-1',
        'IE': 'eu-west-1',
        'GB': 'eu-west-2',
        'FR': 'eu-west-3',
        'JP': 'ap-northeast-1',
        'IN': 'ap-south-1'
    };

    if (request.headers['cloudfront-viewer-country']) {
        const countryCode = request.headers['cloudfront-viewer-country'][0].value;
        const region = countryToRegion[countryCode];

        /**
         * If the viewer's country is not in the list you specify, the request
         * goes to the default S3 bucket you've configured.
         */
        if (region) {
            /**
             * If you've set up OAI, the bucket policy in the destination bucket
             * should allow the OAI GetObject operation, as configured by default
             * for an S3 origin with OAI. Another requirement with OAI is to provide
             * the Region so it can be used for the SIGV4 signature. Otherwise, the
             * Region is not required.
             */
            request.origin.s3.region = region;
            const domainName = `my-bucket-in-${region}.s3.amazonaws.com`;
            request.origin.s3.domainName = domainName;
            request.headers['host'] = [{ key: 'host', value: domainName }];
        }
    }

    callback(null, request);
};
```

Python

```
def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
```

```
    '''
    This blueprint demonstrates how an origin-request trigger can be used to
    change the origin from which the content is fetched, based on request
properties.
    In this example, we use the value of the CloudFront-Viewer-Country header
    to update the S3 bucket domain name to a bucket in a Region that is closer to
    the viewer.

    This can be useful in several ways:
        1) Reduces latencies when the Region specified is nearer to the viewer's
            country.
        2) Provides data sovereignty by making sure that data is served from an
            origin that's in the same country that the request came from.

    NOTE: 1. You must configure your distribution to cache based on the
            CloudFront-Viewer-Country header. For more information, see
            https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
          2. CloudFront adds the CloudFront-Viewer-Country header after the viewer
            request event. To use this example, you must create a trigger for the
            origin request event.
    '''

    countryToRegion = {
        'DE': 'eu-central-1',
        'IE': 'eu-west-1',
        'GB': 'eu-west-2',
        'FR': 'eu-west-3',
        'JP': 'ap-northeast-1',
        'IN': 'ap-south-1'
    }

    viewerCountry = request['headers'].get('cloudfront-viewer-country')
    if viewerCountry:
        countryCode = viewerCountry[0]['value']
        region = countryToRegion.get(countryCode)

        # If the viewer's country in not in the list you specify, the request
        # goes to the default S3 bucket you've configured
        if region:
            '''
            If you've set up OAI, the bucket policy in the destination bucket
            should allow the OAI GetObject operation, as configured by default
            for an S3 origin with OAI. Another requirement with OAI is to provide
            the Region so it can be used for the SIGV4 signature. Otherwise, the
```

```
            Region is not required.
            '''
            request['origin']['s3']['region'] = region
            domainName = 'my-bucket-in-%s.s3.amazonaws.com' % region
            request['origin']['s3']['domainName'] = domainName
            request['headers']['host'] = [{'key': 'host', 'value': domainName}]

    return request
```

例: オリジンリクエストトリガーを使用して Amazon S3 オリジンからカスタムオリジンに変更する

この関数では、origin-request トリガーを使用して、リクエストのプロパティに基づいて、コンテンツがフェッチされるカスタムオリジンを変更する方法を示しています。

Node.js

```
'use strict';

const querystring = require('querystring');

 exports.handler = (event, context, callback) => {
     const request = event.Records[0].cf.request;

     /**
      * Reads query string to check if custom origin should be used, and
      * if true, sets custom origin properties.
      */

     const params = querystring.parse(request.querystring);

     if (params['useCustomOrigin']) {
         if (params['useCustomOrigin'] === 'true') {

             /* Set custom origin fields*/
             request.origin = {
                 custom: {
                     domainName: 'www.example.com',
                     port: 443,
                     protocol: 'https',
                     path: '',
                     sslProtocols: ['TLSv1', 'TLSv1.1'],
                     readTimeout: 5,
```

```
                    keepaliveTimeout: 5,
                    customHeaders: {}
                }
            };
            request.headers['host'] = [{ key: 'host', value: 'www.example.com'}];
        }
    }
    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    # Reads query string to check if custom origin should be used, and
    # if true, sets custom origin properties

    params = {k: v[0] for k, v in parse_qs(request['querystring']).items()}

    if params.get('useCustomOrigin') == 'true':
            # Set custom origin fields
            request['origin'] = {
                'custom': {
                    'domainName': 'www.example.com',
                    'port': 443,
                    'protocol': 'https',
                    'path': '',
                    'sslProtocols': ['TLSv1', 'TLSv1.1'],
                    'readTimeout': 5,
                    'keepaliveTimeout': 5,
                    'customHeaders': {}
                }
            }
            request['headers']['host'] = [{'key': 'host', 'value':
 'www.example.com'}]

    return request
```

例: オリジンリクエストトリガーを使用して Amazon S3 バケットから別のバケットにトラフィックを徐々に転送する

この関数では、Amazon S3 バケットから別のバケットにトラフィックを制御しながら徐々に転送する方法を示しています。

Node.js

```
'use strict';

    function getRandomInt(min, max) {
        /* Random number is inclusive of min and max*/
        return Math.floor(Math.random() * (max - min + 1)) + min;
 }

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const BLUE_TRAFFIC_PERCENTAGE = 80;

    /**
      * This Lambda function demonstrates how to gradually transfer traffic from
      * one S3 bucket to another in a controlled way.
      * We define a variable BLUE_TRAFFIC_PERCENTAGE which can take values from
      * 1 to 100. If the generated randomNumber less than or equal to
 BLUE_TRAFFIC_PERCENTAGE, traffic
      * is re-directed to blue-bucket. If not, the default bucket that we've
 configured
      * is used.
      */

    const randomNumber = getRandomInt(1, 100);

if (randomNumber <= BLUE_TRAFFIC_PERCENTAGE) {
         const domainName = 'blue-bucket.s3.amazonaws.com';
         request.origin.s3.domainName = domainName;
         request.headers['host'] = [{ key: 'host', value: domainName}];
     }
    callback(null, request);
};
```

Python

```
import math
```

```
import random

def getRandomInt(min, max):
    # Random number is inclusive of min and max
    return math.floor(random.random() * (max - min + 1)) + min

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    BLUE_TRAFFIC_PERCENTAGE = 80

    '''
    This Lambda function demonstrates how to gradually transfer traffic from
    one S3 bucket to another in a controlled way.
    We define a variable BLUE_TRAFFIC_PERCENTAGE which can take values from
    1 to 100. If the generated randomNumber less than or equal to
 BLUE_TRAFFIC_PERCENTAGE, traffic
    is re-directed to blue-bucket. If not, the default bucket that we've configured
    is used.
    '''

    randomNumber = getRandomInt(1, 100)

    if randomNumber <= BLUE_TRAFFIC_PERCENTAGE:
        domainName = 'blue-bucket.s3.amazonaws.com'
        request['origin']['s3']['domainName'] = domainName
        request['headers']['host'] = [{'key': 'host', 'value': domainName}]

    return request
```

例: オリジンリクエストトリガーを使用して Country ヘッダーに基づいてオリジンのドメイン名を変更する

この関数では、CloudFront-Viewer-Country ヘッダーに基づいてオリジンのドメイン名を変更する方法を示しています。これにより、コンテンツはビューワーの国に近いオリジンから配信されます。

ディストリビューションに対してこの機能を実装すると、次のような利点があります。

- 指定したリージョンがビューワーの国に近いほど、レイテンシーが短縮されます。
- リクエスト元と同じ国にあるオリジンからデータが提供されることになり、データ主権が確保されます。

この機能を有効にするには、CloudFront-Viewer-Country ヘッダーに基づいてキャッシュするようにディストリビューションを設定する必要があります。詳細については、「the section called "選択されたリクエストヘッダーに基づいたキャッシュ"」を参照してください。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
     const request = event.Records[0].cf.request;

  if (request.headers['cloudfront-viewer-country']) {
        const countryCode = request.headers['cloudfront-viewer-country'][0].value;
        if (countryCode === 'GB' || countryCode === 'DE' || countryCode === 'IE' )
 {
            const domainName = 'eu.example.com';
            request.origin.custom.domainName = domainName;
            request.headers['host'] = [{key: 'host', value: domainName}];
        }
     }

    callback(null, request);
};
```

Python

```
def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    viewerCountry = request['headers'].get('cloudfront-viewer-country')
    if viewerCountry:
        countryCode = viewerCountry[0]['value']
        if countryCode == 'GB' or countryCode == 'DE' or countryCode == 'IE':
            domainName = 'eu.example.com'
            request['origin']['custom']['domainName'] = domainName
            request['headers']['host'] = [{'key': 'host', 'value': domainName}]
    return request
```

エラーステータスを更新する - 例

このセクションの例では、Lambda@Edge を使用して、ユーザーに返されるエラーステータスを変更する方法を示しています。

トピック

- 例: オリジンレスポンストリガーを使用してエラーステータスコードを 200 に更新する
- 例: オリジンレスポンストリガーを使用してエラーステータスコードを 302 に更新する

例: オリジンレスポンストリガーを使用してエラーステータスコードを 200 に更新する

この関数では、レスポンスステータスを 200 に更新し、以下のシナリオでビューワーに返す静的な本文コンテンツを生成する方法を示しています。

- 関数がオリジンレスポンスでトリガーされる。
- オリジンサーバーからのレスポンスステータスがエラーステータスコード (4xx または 5xx) である。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const response = event.Records[0].cf.response;

    /**
     * This function updates the response status to 200 and generates static
     * body content to return to the viewer in the following scenario:
     * 1. The function is triggered in an origin response
     * 2. The response status from the origin server is an error status code (4xx or
 5xx)
     */

    if (response.status >= 400 && response.status <= 599) {
        response.status = 200;
        response.statusDescription = 'OK';
        response.body = 'Body generation example';
    }

    callback(null, response);
```

```
};
```

Python

```
def lambda_handler(event, context):
    response = event['Records'][0]['cf']['response']

    '''
    This function updates the response status to 200 and generates static
    body content to return to the viewer in the following scenario:
    1. The function is triggered in an origin response
    2. The response status from the origin server is an error status code (4xx or
 5xx)
    '''

    if int(response['status']) >= 400 and int(response['status']) <= 599:
        response['status'] = 200
        response['statusDescription'] = 'OK'
        response['body'] = 'Body generation example'
    return response
```

例: オリジンレスポンストリガーを使用してエラーステータスコードを 302 に更新する

この関数では、HTTP ステータスコードを 302 に更新して、異なるオリジンを設定した別のパス (キャッシュ動作) にリダイレクトする方法を示しています。次の点に注意してください。

- 関数がオリジンレスポンスでトリガーされる。
- オリジンサーバーからのレスポンスステータスがエラーステータスコード (4xx または 5xx) である。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const response = event.Records[0].cf.response;
    const request = event.Records[0].cf.request;

    /**
     * This function updates the HTTP status code in the response to 302, to
 redirect to another
```

```
    * path (cache behavior) that has a different origin configured. Note the
 following:
    * 1. The function is triggered in an origin response
    * 2. The response status from the origin server is an error status code (4xx or
 5xx)
    */

   if (response.status >= 400 && response.status <= 599) {
       const redirect_path = `/plan-b/path?${request.querystring}`;

       response.status = 302;
       response.statusDescription = 'Found';

       /* Drop the body, as it is not required for redirects */
       response.body = '';
       response.headers['location'] = [{ key: 'Location', value: redirect_path }];
   }

   callback(null, response);
};
```

Python

```
def lambda_handler(event, context):
    response = event['Records'][0]['cf']['response']
    request = event['Records'][0]['cf']['request']

    '''
    This function updates the HTTP status code in the response to 302, to redirect
 to another
    path (cache behavior) that has a different origin configured. Note the
 following:
    1. The function is triggered in an origin response
    2. The response status from the origin server is an error status code (4xx or
 5xx)
    '''

    if int(response['status']) >= 400 and int(response['status']) <= 599:
        redirect_path = '/plan-b/path?%s' % request['querystring']

        response['status'] = 302
        response['statusDescription'] = 'Found'
```

```
        # Drop the body as it is not required for redirects
        response['body'] = ''
        response['headers']['location'] = [{'key': 'Location', 'value':
redirect_path}]

    return response
```

リクエストボディにアクセスする - 例

このセクションの例では、Lambda@Edge を使用して POST リクエストを操作する方法を示しています。

> **Note**
>
> これらの例を使用するには、ディストリビューションの Lambda 関数の関連付けで [Include body] (ボディを含める) オプションを有効にする必要があります。デフォルトでは、有効になっていません。
>
> - CloudFront コンソールでこの設定を有効にするには、[Lambda 関数の関連付け] の [ボディを含める] チェックボックスをオンにします。
> - CloudFront API または AWS CloudFormation でこの設定を有効にするには、`LambdaFunctionAssociation` の `IncludeBody` フィールドを `true` に設定します。

トピック

- 例: リクエストトリガーを使用して HTML フォームを読み込む
- 例: リクエストトリガーを使用して HTML フォームを変更する

例: リクエストトリガーを使用して HTML フォームを読み込む

この関数では、「お問い合わせ」フォームなど HTML フォーム (ウェブフォーム) によって生成された POST リクエストボディを処理する方法を示しています。たとえば、次のような HTML フォームがあります。

```
<html>
  <form action="https://example.com" method="post">
```

```
    Param 1: <input type="text" name="name1"><br>
    Param 2: <input type="text" name="name2"><br>
    input type="submit" value="Submit">
  </form>
</html>
```

次の関数の例では、関数は CloudFront ビューワーリクエストまたはオリジンリクエストでトリガーされる必要があります。

Node.js

```
'use strict';

const querystring = require('querystring');

/**
 * This function demonstrates how you can read the body of a POST request
 * generated by an HTML form (web form). The function is triggered in a
 * CloudFront viewer request or origin request event type.
 */

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;

    if (request.method === 'POST') {
        /* HTTP body is always passed as base64-encoded string. Decode it. */
        const body = Buffer.from(request.body.data, 'base64').toString();

        /* HTML forms send the data in query string format. Parse it. */
        const params = querystring.parse(body);

        /* For demonstration purposes, we only log the form fields here.
         * You can put your custom logic here. For example, you can store the
         * fields in a database, such as Amazon DynamoDB, and generate a response
         * right from your Lambda@Edge function.
         */
        for (let param in params) {
            console.log(`For "${param}" user submitted "${params[param]}".\n`);
        }
    }
    return callback(null, request);
};
```

Python

```
import base64
from urllib.parse import parse_qs

'''
Say there is a POST request body generated by an HTML such as:

<html>
<form action="https://example.com" method="post">
    Param 1: <input type="text" name="name1"><br>
    Param 2: <input type="text" name="name2"><br>
    input type="submit" value="Submit">
</form>
</html>

'''

'''
This function demonstrates how you can read the body of a POST request
generated by an HTML form (web form). The function is triggered in a
CloudFront viewer request or origin request event type.
'''

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    if request['method'] == 'POST':
        # HTTP body is always passed as base64-encoded string. Decode it
        body = base64.b64decode(request['body']['data'])

        # HTML forms send the data in query string format. Parse it
        params = {k: v[0] for k, v in parse_qs(body).items()}

        '''
        For demonstration purposes, we only log the form fields here.
        You can put your custom logic here. For example, you can store the
        fields in a database, such as Amazon DynamoDB, and generate a response
        right from your Lambda@Edge function.
        '''
        for key, value in params.items():
            print("For %s use submitted %s" % (key, value))
```

```
    return request
```

例: リクエストトリガーを使用して HTML フォームを変更する

この関数では、HTML フォーム (ウェブフォーム) によって生成された POST リクエストボディを変更する方法を示しています。この関数は、CloudFront ビューワーリクエストまたはオリジンリクエストでトリガーされます。

Node.js

```
'use strict';

const querystring = require('querystring');

exports.handler = (event, context, callback) => {
    var request = event.Records[0].cf.request;
    if (request.method === 'POST') {
        /* Request body is being replaced. To do this, update the following
        /* three fields:
         *    1) body.action to 'replace'
         *    2) body.encoding to the encoding of the new data.
         *
         *       Set to one of the following values:
         *
         *           text - denotes that the generated body is in text format.
         *               Lambda@Edge will propagate this as is.
         *           base64 - denotes that the generated body is base64 encoded.
         *               Lambda@Edge will base64 decode the data before sending
         *               it to the origin.
         *    3) body.data to the new body.
         */
        request.body.action = 'replace';
        request.body.encoding = 'text';
        request.body.data = getUpdatedBody(request);
    }
    callback(null, request);
};

function getUpdatedBody(request) {
    /* HTTP body is always passed as base64-encoded string. Decode it. */
    const body = Buffer.from(request.body.data, 'base64').toString();
```

```
    /* HTML forms send data in query string format. Parse it. */
    const params = querystring.parse(body);

    /* For demonstration purposes, we're adding one more param.
     *
     * You can put your custom logic here. For example, you can truncate long
     * bodies from malicious requests.
     */
    params['new-param-name'] = 'new-param-value';
    return querystring.stringify(params);
}
```

Python

```
import base64
from urllib.parse import parse_qs, urlencode

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    if request['method'] == 'POST':
        '''
        Request body is being replaced. To do this, update the following
        three fields:
            1) body.action to 'replace'
            2) body.encoding to the encoding of the new data.

            Set to one of the following values:

                text - denotes that the generated body is in text format.
                    Lambda@Edge will propagate this as is.
                base64 - denotes that the generated body is base64 encoded.
                    Lambda@Edge will base64 decode the data before sending
                    it to the origin.
            3) body.data to the new body.
        '''
        request['body']['action'] = 'replace'
        request['body']['encoding'] = 'text'
        request['body']['data'] = getUpdatedBody(request)
    return request

def getUpdatedBody(request):
    # HTTP body is always passed as base64-encoded string. Decode it
    body = base64.b64decode(request['body']['data'])
```

```
    # HTML forms send data in query string format. Parse it
    params = {k: v[0] for k, v in parse_qs(body).items()}

    # For demonstration purposes, we're adding one more param

    # You can put your custom logic here. For example, you can truncate long
    # bodies from malicious requests
    params['new-param-name'] = 'new-param-value'
    return urlencode(params)
```

エッジ関数に対する制限

以下のトピックでは、CloudFront Functions と Lambda@Edge に適用される制限について説明します。制限には、すべてのエッジ関数に適用されるものもあれば、CloudFront Functions または Lambda@Edge のみに適用されるものもあります。

クォータ (以前は制限と呼ばれていました) の詳細については、「CloudFront Functions のクォータ」と「Lambda@Edge のクォータ」を参照してください。

トピック

- すべてのエッジ機能に対する制限
- CloudFront Functions に対する制限
- Lambda@Edge に対する制限

すべてのエッジ機能に対する制限

以下の制限は、CloudFront Functions と Lambda@Edge 両方の、すべてのエッジ関数に適用されます。

トピック

- AWS アカウント の所有権
- CloudFront Functions と Lambda@Edge との組み合わせ
- HTTP ステータスコード
- HTTP ヘッダー
- クエリ文字列

- [URI]
- URI とクエリ文字列のエンコーディング
- Microsoft Smooth Streaming
- タグ付け

AWS アカウント の所有権

エッジ関数を CloudFront ディストリビューションに関連付けるには、関数とディストリビューションが同じ AWS アカウント によって所有されている必要があります。

CloudFront Functions と Lambda@Edge との組み合わせ

所定のキャッシュ動作については、以下の制限が適用されます。

- イベントタイプ (ビューワーリクエスト、オリジンリクエスト、オリジンレスポンス、ビューワーレスポンス) はそれぞれ、エッジ関数の関連付けを 1 つしか持てません。
- ビューワーイベント (ビューワーリクエストとビューワーレスポンス) で CloudFront Functions と Lambda@Edge を組み合わせることはできません。

上記以外のすべてのエッジ関数の組み合わせが許可されます。以下の表は、許可される組み合わせの説明です。

		CloudFront Functions	
		ビューワーリクエスト	ビューワーレスポンス
Lambda@Edge	ビューワーリクエスト	許可されていません	許可されていません
	オリジンリクエスト	許可	許可
	オリジンレスポンス	許可	許可
	ビューワーレスポンス	許可されていません	許可されていません

HTTP ステータスコード

オリジンが 400 以上の HTTP ステータスコードを返す場合、CloudFront はビューワーレスポンスイベントの Lambda 関数を呼び出しません。

オリジンレスポンスイベントの Lambda@Edge 関数は、オリジンが 400 以上の HTTP ステータスコードを返す場合を含め、すべてのオリジンレスポンスに対して呼び出されます。詳細については、「オリジンレスポンストリガーでの HTTP レスポンスを更新する」を参照してください。

HTTP ヘッダー

特定の HTTP ヘッダーは許可されていません。これは、これらのヘッダーが工ッジ関数に公開されておらず、関数がそれらを追加できないことを意味します。他のヘッダーは読み取り専用です。つまり、関数はこれらを読み取れますが、追加したり変更したりすることはできません。

トピック

- 許可されていないヘッダー
- 読み取り専用ヘッダー

許可されていないヘッダー

以下の HTTP ヘッダーは工ッジ関数に公開されておらず、関数はこれらを追加できません。関数がこれらのヘッダーのいずれかを追加すると、CloudFront 検証が失敗し、CloudFront がビューワーに HTTP ステータスコード 502 (不正なゲートウェイ) を返します。

- `Connection`
- `Expect`
- `Keep-Alive`
- `Proxy-Authenticate`
- `Proxy-Authorization`
- `Proxy-Connection`
- `Trailer`
- `Upgrade`
- `X-Accel-Buffering`
- `X-Accel-Charset`

- X-Accel-Limit-Rate
- X-Accel-Redirect
- X-Amz-Cf-*
- X-Amzn-Auth
- X-Amzn-Cf-Billing
- X-Amzn-Cf-Id
- X-Amzn-Cf-Xff
- X-Amzn-Errortype
- X-Amzn-Fle-Profile
- X-Amzn-Header-Count
- X-Amzn-Header-Order
- X-Amzn-Lambda-Integration-Tag
- X-Amzn-RequestId
- X-Cache
- X-Edge-*
- X-Forwarded-Proto
- X-Real-IP

読み取り専用ヘッダー

以下のヘッダーは読み取り専用です。関数はこれらを読み取って関数ロジックへの入力として使用できますが、値を変更することはできません。関数が読み取り専用ヘッダーを追加または編集すると、リクエストの CloudFront 検証が失敗し、CloudFront がビューワーに HTTP ステータスコード 502 (不正なゲートウェイ) を返します。

ビューワーリクエストイベントの読み取り専用ヘッダー

以下のヘッダーは、ビューワーリクエストイベントでは読み取り専用になります。

- Content-Length
- Host
- Transfer-Encoding
- Via

オリジンリクエストイベントの読み取り専用ヘッダー (Lambda@Edge 限定)

以下のヘッダーは、Lambda@Edge にしかないオリジンリクエストイベントでは読み取り専用になります。

- `Accept-Encoding`
- `Content-Length`
- `If-Modified-Since`
- `If-None-Match`
- `If-Range`
- `If-Unmodified-Since`
- `Transfer-Encoding`
- `Via`

オリジンレスポンスイベントの読み取り専用ヘッダー (Lambda@Edge 限定)

以下のヘッダーは、Lambda@Edge にしかないオリジンレスポンスイベントでは読み取り専用になります。

- `Transfer-Encoding`
- `Via`

ビューワーレスポンスイベントの読み取り専用ヘッダー

以下のヘッダーは、CloudFront Functions と Lambda@Edge の両方のビューワーレスポンスイベントでは読み取り専用になります。

- `Warning`
- `Via`

以下のヘッダーは、Lambda@Edge のビューワーレスポンスイベントでは読み取り専用になります。

- `Content-Length`
- `Content-Encoding`

- `Transfer-Encoding`

クエリ文字列

以下の制限は、リクエスト URI 内のクエリ文字列を読み取る、更新する、または作成する関数に適用されます。

- (Lambda@Edge 限定) オリジンリクエストまたはオリジンレスポンス関数のクエリ文字列にアクセスするには、キャッシュポリシーまたはオリジンリクエストポリシーが [Query strings] (クエリ文字列) に対して [All] (すべて) に設定されている必要があります。
- 関数は、ビューワーリクエストイベントとオリジンリクエストイベントのクエリ文字列を作成または更新できます (オリジンリクエストイベントがあるのは Lambda@Edge だけです)。
- 関数は、オリジンレスポンスイベントとビューワーレスポンスイベントのクエリ文字列を読み取ることができますが、それらを作成または更新することはできません (オリジンレスポンスイベントがあるのは Lambda@Edge だけです)。
- 関数がクエリ文字列を作成または更新する場合は、以下の制限が適用されます。
  - クエリ文字列に、スペース、制御文字、またはフラグメント識別子 (#) を含めることはできません。
  - クエリ文字列を含めた URI の合計サイズは、8,192 文字未満にする必要があります。
  - URI およびクエリ文字列には、パーセントエンコーディングを使用することをお勧めします。詳細については、「URI とクエリ文字列のエンコーディング」を参照してください。

[URI]

関数でリクエストの URI を変更しても、リクエストのキャッシュ動作や転送先オリジンは変わりません。

クエリ文字列を含めた URI の合計サイズは、8,192 文字未満にする必要があります。

URI とクエリ文字列のエンコーディング

エッジ関数に渡される URI とクエリ文字列値は、UTF-8 でエンコードされています。関数は、それが返す URI とクエリ文字列値に UTF-8 エンコーディングを使用する必要があります。パーセントエンコーディングには、UTF-8 エンコーディングと互換性があります。

以下のリストでは、CloudFront が URI とクエリ文字列値のエンコーディングをどのように処理するかを説明します。

- リクエスト内の値が UTF-8 でエンコードされている場合、CloudFront はそれらの値を変更せずにそのまま関数に転送します。
- リクエスト内の値が ISO 8859-1 でエンコードされている場合、CloudFront はそれらの値を UTF-8 エンコーディングに変換してから関数に転送します。
- リクエスト内の値がその他の文字エンコーディングを使用してエンコードされている場合、CloudFront はそれらが ISO 8859-1 でエンコードされているとみなし、ISO 8859-1 から UTF-8 への変換を試みます。

> **Important**
>
> 変換された文字は、元のリクエストの値の正しい解釈ではない可能性があります。これは、関数またはオリジンが意図しない結果を生成する原因になる場合があります。

CloudFront がオリジンに転送する URI とクエリ文字列値は、関数がそれらの値を変更するかどうかに応じて異なります。

- 関数が URI またはクエリ文字列を変更しない場合、CloudFront はリクエストで受け取った値をオリジンに転送します。
- 関数が URI またはクエリ文字列を変更する場合、CloudFront は UTF-8 でエンコードされた値を転送します。

Microsoft Smooth Streaming

Microsoft Smooth Streaming 形式にトランスコードしたメディアファイルのストリーミングに使用している CloudFront ディストリビューションでエッジ関数を使用することはできません。

タグ付け

エッジ関数にタグを追加することはできません。CloudFront でのタグ付けの詳細については、「ディストリビューションのタグ付け」を参照してください。

CloudFront Functions に対する制限

以下の制限は、CloudFront Functions のみに適用されます。

クォータ (以前は制限と呼ばれていました) の詳細については、「CloudFront Functions のクォータ」を参照してください。

ログ

CloudFront Functions の関数ログは 10 KB で切り捨てられます。

リクエストボディ

CloudFront Functions は、HTTP リクエストのボディにアクセスできません。

CloudFront KeyValueStore API を使用する場合のリージョン AWS Security Token Service エンドポイント

一時的なセキュリティ認証情報で署名バージョン 4A (SigV4A) を使用して CloudFront KeyValueStore API を呼び出す場合 (AWS Identity and Access Management (IAM) ロールを使用する場合など) は、必ず AWS STS でリージョンエンドポイントからの一時的認証情報をリクエストしてください。AWS STS (`sts.amazonaws.com`) でグローバルエンドポイントを使用すると、AWS STS はグローバルエンドポイントから一時的な認証情報を生成しますが、これは Sig4A ではサポートされません。その結果、認証エラーが発生します。この問題を解決するには、「IAM ユーザーガイド」にリストされている「AWS STS 用のリージョンエンドポイント」のいずれかを使用します。AWS STS リージョンエンドポイントを使用するように SAML を設定する場合は、ブログ記事「フェイルオーバーにリージョン SAML エンドポイントを使用する方法」を参照してください。

ランタイム

CloudFront Functions のランタイム環境は動的コード評価をサポートせず、ネットワーク、ファイルシステム、およびタイマーへのアクセスを制限します。詳細については、「制限された機能」を参照してください。

> **Note**
>
> CloudFront KeyValueStore を使用するには、CloudFront 関数で JavaScript ランタイム 2.0 を使用する必要があります。

コンピューティング使用率

CloudFront Functions には実行に使用できる時間に対する制限があり、これはコンピューティング使用率として測定されます。コンピューティング使用率は、関数の実行にかかった時間を最大許容時間に対する割合として示す 0 から 100 の数値です。例えば、コンピューティング使用率が 35 の場合、関数は最大許容時間の 35% で完了したことを意味します。

関数をテストするときは、テストイベントの出力でコンピューティング使用率の値を確認できます。production 関数については、CloudWatch、または CloudFront コンソールの [Monitoring] (モニタリング) ページでコンピューティング使用率メトリクスを確認できます。

Lambda@Edge に対する制限

以下の制限は、Lambda@Edge のみに適用されます。

クォータの詳細については、「Lambda@Edge のクォータ」を参照してください。

DNS 解決

CloudFront は、オリジンドメイン名で DNS 解決を実行してから、オリジンリクエストの Lambda@Edge 関数を実行します。ドメインの DNS サービスで問題が発生しているために、CloudFront がドメイン名の解決によって IP アドレスを取得できない場合、Lambda@Edge 関数は呼び出されません。CloudFront は、HTTP 502 ステータスコード (不正なゲートウェイ) をクライアントに返します。詳細については、「DNS エラー (NonS3OriginDnsError) 」を参照してください。

DNS フェイルオーバーの管理の詳細については、「Amazon Route 53 デベロッパーガイド」の「DNS フェイルオーバーの設定」を参照してください。

HTTP ステータスコード

ビューワーレスポンスイベントの Lambda@Edge 関数は、レスポンスがオリジンまたは CloudFront キャッシュからのものであるかどうかにかかわらず、レスポンスの HTTP ステータスコードを変更できません。

Lambda 関数のバージョン

`$LATEST` やエイリアスではなく、Lambda 関数の番号付きバージョンを使用する必要があります。

Lambda リージョン

Lambda 関数は、米国東部 (バージニア北部) リージョンにある必要があります。

Lambda のロール許可

Lambda 関数に関連付けられている IAM 実行ロールは、サービスプリンシパル `lambda.amazonaws.com` と `edgelambda.amazonaws.com` によるそのロールの引き受けを許可

する必要があります。詳細については、「Lambda@Edge 用の IAM アクセス許可とロールのセットアップ」を参照してください。

Lambda の機能

以下の Lambda 機能は、Lambda@Edge でサポートされていません。

- [自動] (デフォルト) 以外の Lambda ランタイム管理設定
- VPC 内のリソースにアクセスするための Lambda 関数の設定
- Lambda 関数のデッドレターキュー
- Lambda 環境変数 (自動的にサポートされる予約環境変数は除く)
- AWS Lambda レイヤーを使用する Lambda 関数
- AWS X-Ray を使用する
- Lambda プロビジョニング済み同時実行

> **Note**
>
> Lambda@Edge 関数には、Lambda 関数と同じリージョンの同時実行機能があります。ただし、Lambda@Edge の同時実行数のクォータを増やすと、Lambda@Edge 関数をレプリケートしたすべての AWS リージョンでクォータが増えます。詳細については、「Lambda@Edge のクォータ」を参照してください。

- コンテナイメージとして定義した Lambda 関数
- arm64 アーキテクチャを使用する Lambda 関数
- エフェメラルストレージが 512 MB を超える Lambda 関数
- JSON 構造化形式での Lambda 関数ログのキャプチャ
- Lambda 関数ログのログレベルの詳細度の制御
- Lambda がログを送信する Amazon CloudWatch ロググループの設定

ランタイムのサポート

Lambda@Edge は、以下のランタイムで Lambda 関数をサポートします。

Node.js	Python
• Node.js 20	• Python 3.12

Node.js	Python
• Node.js 18 • Node.js 16[1] • Node.js 14[2] • Node.js 12[2] • Node.js 10[2] • Node.js 8[2] • Node.js 6[2]	• 「Python 3.11」 • 「Python 3.10」 • Python 3.9 • Python 3.8 • Python 3.7

[1]このバージョンの Node.js はサポートが終了し、AWS Lambda によってまもなく廃止されます。

[2]このバージョンの Node.js はサポートが終了し、AWS Lambda によって完全に廃止されました。

廃止されたバージョンの Node.js で関数を作成または更新することはできません。これらのバージョンの既存の関数は、CloudFront ディストリビューションにのみ関連付けることができます。これらのバージョンに該当する関数のうち、ディストリビューションに関連付けられているものは、引き続き実行されます。ただし、関数を新しいバージョンの Node.js に移行することをお勧めします。詳細については、「AWS Lambda デベロッパーガイド」の「ランタイムの非推奨化に関するポリシー」と、GitHub の Node.js リリーススケジュールを参照してください。

> **Tip**
>
> ベストプラクティスとして、提供されている最新バージョンのランタイムでパフォーマンスを改善し、新しい機能を使用してください。

CloudFront ヘッダー

Lambda@Edge 関数は、CloudFront のリクエストヘッダーを追加する にリストされている任意の CloudFront ヘッダーの読み取り、編集、削除、または追加を行うことができます。

> **メモ**
>
> • これらのヘッダーを CloudFront で追加する場合は、キャッシュポリシーまたはオリジンリクエストポリシーを使用してヘッダーを追加するように CloudFront を設定する必要があります。

- CloudFront は、ビューワーリクエストイベントの後にヘッダーを追加します。つまり、ビューワーリクエストの Lambda@Edge 関数ではヘッダーを使用できません。ヘッダーは、オリジンリクエストとオリジンレスポンスの Lambda@Edge 関数でのみ使用できます。
- ビューワーリクエストにこれらの名前を持つヘッダーが含まれており、キャッシュポリシーまたはオリジンリクエストポリシーを使用してこれらのヘッダーを追加するように CloudFront を設定した場合は、CloudFront がビューワーリクエストにあったヘッダー値を上書きします。ビューワー向けの関数はビューワーリクエストからのヘッダー値を認識し、オリジン向けの関数は CloudFront が追加したヘッダー値を認識します。
- ビューワーリクエスト関数で `CloudFront-Viewer-Country` ヘッダーを追加すると、検証に失敗し、CloudFront は HTTP ステータスコード 502 (不正なゲートウェイ) をビューワーに返します。

Include Body オプションがあるリクエストボディに対する制限

Lambda@Edge 関数にリクエストボディを公開するために [Include Body] オプションを選択する場合は、公開または置き換えられたボディの一部に以下の情報クォータとサイズクォータが適用されます。

- CloudFront は常に、リクエストボディを base64 でエンコードしてから、それを Lambda@Edge に公開します。
- リクエストボディが大きい場合、CloudFront は、ボディを以下のように切り詰めてから Lambda@Edge に公開します。
  - ビューワーリクエストでは、ボディが 40 KB で切り捨てられます。
  - オリジンリクエストでは、ボディが 1 MB で切り捨てられます。
- 読み取り専用としてリクエストボディにアクセスする場合、CloudFront は元の完全なリクエストボディをオリジンに送信します。
- Lambda@Edge 関数がリクエストボディを置き換える場合、関数が返すボディに以下のサイズクォータが適用されます。
  - Lambda@Edge 関数がボディをプレーンテキストとして返す場合:
    - ビューワーリクエストでは、ボディが 40 KB で切り捨てられます。
    - オリジンリクエストでは、ボディが 1 MB で切り捨てられます。
  - Lambda@Edge 関数がボディを base64 でエンコードされたテキストとして返す場合:
    - ビューワーリクエストイベントでは、ボディが 53.2 KB で切り捨てられます。

- オリジンリクエストイベントでは、ボディが 1.33 MB で切り捨てられます。

レスポンスタイムアウトとキープアライブタイムアウト (カスタムオリジンのみ)

Lambda@Edge 関数を使用してディストリビューションオリジンのレスポンスタイムアウトまたはキープアライブタイムアウトを設定している場合は、オリジンがサポートできる値を指定していることを確認します。詳細については、「レスポンスとキープアライブのタイムアウトクォータ」を参照してください。

レポート、メトリクス、ログ

CloudFront には、CloudFront リソースのレポート、モニタリング、ログ記録のためのオプションがいくつか用意されています。

- レポートを表示およびダウンロードして、請求レポート、キャッシュ統計、人気の高いコンテンツ、上位リファラなど、CloudFront ディストリビューションの使用状況とアクティビティを確認できます。
- CloudFront コンソールで直接、または Amazon CloudWatch を使用して、エッジコンピューティング関数を含む CloudFront をモニタリングおよび追跡できます。CloudFront は、Lambda@Edge 関数と CloudFront Functions の両方で、ディストリビューション関数とエッジ関数のさまざまなメトリクスを CloudWatch に送信します。
- CloudFront ディストリビューションが標準ログまたはリアルタイム ログで受け取るビューワーリクエストのログを表示できます。ビューワーリクエストログに加えて、CloudWatch Logs を使用して、Lambda@Edge 関数と CloudFront Functions の両方のエッジ関数のログを取得できます。AWS CloudTrail を使用して、AWS アカウント で CloudFront API アクティビティのログを取得することもできます。
- AWS Config を使用して、CloudFront リソースへの設定変更を追跡できます。

これらのオプションのそれぞれの詳細については、以下のトピックを参照してください。

トピック

CloudFront の AWS の請求書と使用状況レポート

AWS には、CloudFront の使用状況レポートが 2 種類、用意されています。

- AWS 請求レポートは、CloudFront も含めて、使用している AWS のサービスのアクティビティすべての概要を示します。

- AWS 使用状況レポートは、特定のサービスのアクティビティの概要を、時間、日、または月単位で集約して示します。CloudFront の使用状況をグラフィカルに表現する使用状況グラフも含まれています。

> **Note**
>
> 他の AWS のサービスと同様に、CloudFront は実際に使用した分のみの料金請求となります。詳細については、「CloudFront の料金」を参照してください。

トピック

CloudFront の AWS 請求レポートを表示する

AWS Billing and Cost Management コンソールの [請求書] ページで、AWS の使用状況と料金の概要をサービス別に表示できます。

AWS 請求レポートを表示するには

1. AWS Management Console にサインインして AWS Billing コンソール (https://console.aws.amazon.com/billing/) を開きます。
2. ナビゲーションペインで [Bills (請求書)] を選択します。
3. [請求期間] (例: 2023 年 8 月) を選択します。
4. [サービス別料金] タブで、[CloudFront]を選択し、[グローバル] または AWS リージョン の名前を展開します。
5. 詳細な請求レポートを CSV 形式でダウンロードするには、[すべてを CSV にダウンロード] を選択します。

AWS 請求書の詳細については、「AWS Billing ユーザーガイド」の「請求書を表示する」を参照してください。

請求レポートには、CloudFront に該当する以下の値が表示されます。

- ProductCode — AmazonCloudFront
- 使用状況 – 以下の値のいずれかです。
  - データ転送のタイプを識別するコード
  - Invalidations
  - Executions-CloudFrontFunctions
  - KeyValueStore-APIOperations
  - KeyValueStore-EdgeReads
  - RealTimeLog-KinesisDataStream
  - SSL-Cert-Custom
- ItemDescription — UsageType の請求レートの説明。
- UsageStart Date と UsageEndDate – 使用状況が該当する協定世界時 (UTC) の日付。
- UsageQuantity — 以下の値のいずれかです。
  - 指定した期間のリクエストの数
  - データ転送量 (GB)
  - 無効にされたオブジェクトの数
  - SSL 証明書を有効な CloudFront ディストリビューションに関連付けた月数を按分計算した合計値。たとえば、ある証明書を有効なディストリビューションに 1 か月まるまる関連付け、別の証明書を有効なディストリビューションに半月だけ関連付けた場合、この値は 1.5 になります。

CloudFront の AWS 使用状況レポートを表示する

AWS には、CloudFront 使用状況レポートが用意されています。これは請求レポートよりは詳細ですが、CloudFront アクセスログほど詳細ではありません。使用状況レポートには、使用状況データが時間、日、または月単位で集計され、リージョンと使用タイプ別に操作が一覧で示されます (たとえば、オーストラリアリージョンからデータが転送されたなど)。

AWS 使用状況レポートを表示するには

1. AWS Management Console にサインインして AWS Billing コンソール (https://console.aws.amazon.com/billing/) を開きます。
2. ナビゲーションペインで、[コストとレポート] を選択します。
3. [AWS 使用状況レポート]セクションで、[使用状況レポートの作成] を選択します。

4. [使用状況レポートをダウンロード] ページの [サービス] で、[Amazon CloudFront] を選択します。
5. 使用タイプを選択します。
6. [操作]を選択します。
7. レポートの期間を選択します。カスタム日付範囲を選択した場合は、レポートの日付範囲を手動で指定する必要があります。
8. [レポートの精度] で、[時間単位]、[日単位]、または [月単位] を選択します。
9. [ダウンロード] を選択し、[XML レポート] または [CSV レポート] を選択します。

AWS 使用状況レポートの詳細については、「AWS Data Exports ユーザーガイド」の「AWS 使用状況レポート」を参照してください。

CloudFront 使用状況レポートには次の値が含まれています。

- サービス – `AmazonCloudFront`
- オペレーション — HTTP メソッド。値には、`DELETE`、`GET`、`HEAD`、`OPTIONS`、`PATCH`、`POST`、`PUT` があります。
- UsageType – 以下の値のいずれかです。
  - データ転送のタイプを識別するコード
  - `Invalidations`
  - `Executions-CloudFrontFunctions`
  - `KeyValueStore-APIOperations`
  - `KeyValueStore-EdgeReads`
  - `RealTimeLog-KinesisDataStream`
  - `SSL-Cert-Custom`
- リソース — 使用状況に関連付けられた CloudFront ディストリビューションの ID、または、CloudFront ディストリビューションに関連付けた SSL 証明書の証明書 ID。
- StartTime/EndTime — 使用状況が該当する協定世界時 (UTC) の日付。
- UsageValue – 1) 指定した期間のリクエストの数、または、2) 転送データ量 (バイト)。

Amazon S3 を CloudFront のオリジンとして使用している場合は、Amazon S3 の使用状況レポートも作成することを検討してください。ただし、CloudFront ディストリビューションのオリジンとし

て以外の目的でも Amazon S3 を使用している場合は、どの部分が CloudFront で使用されたのか明確にならない可能性があります。

> **Tip**
>
> お客様のディストリビューションの CloudFront アクセスログをオンにすることで、お客様のオブジェクトについて CloudFront が受信するすべてのリクエストに関する詳細な情報を取得することができます。詳細については、「the section called “標準ログ (アクセスログ) の使用”」を参照してください。

レポートの CloudFront の料金と使用タイプを理解するための詳細情報については、「the section called “CloudFront の AWS 請求レポートと使用状況レポートを解釈する”」を参照してください。

CloudFront の AWS 請求レポートと使用状況レポートを解釈する

請求レポートと使用状況レポートを取得したら、このトピックを参考にして、請求書に表示される CloudFront の各料金および対応する使用タイプを解釈する方法を理解できます。このトピックには、両方のレポートに表示されるコードと AWS リージョンの略語が含まれています。

2 つの列のコードには、たいてい、アクティビティの場所を示す 2 文字の略語が入っています。次の表のコードの *region* は、AWS 請求書と使用状況レポートでは以下の 2 文字の略語に置き換えられます。

- AP: 香港、フィリピン、韓国、シンガポール、台湾、シンガポール (アジアパシフィック)
- AU: オーストラリア
- CA: カナダ
- EU: ヨーロッパおよびイスラエル
- IN: インド
- JP: 日本
- ME: 中東
- SA: 南米
- US: 米国
- ZA: 南アフリカ

AWS リージョン別の料金の詳細については、「Amazon CloudFront の料金」を参照してください。

メモ

- 次の表には、Amazon S3 バケットから CloudFront エッジロケーションへのオブジェクト転送の料金は含まれていません。この料金が発生している場合、AWS 請求書の [AWS Data Transfer] (データ転送) の部分に料金が表記されます。
- 最初の列は、AWS 請求レポートに表示される料金と、各料金の明細を示しています。
- 2 列目は、AWS 使用状況レポートに表示される項目と、請求書の料金および使用状況レポートの項目との相互関係を示しています。

AWS 請求書の CloudFront 料金	AWS 使用状況レポートの UsageType 列の値
region-DataTransfer-Out-Bytes ユーザーの GET と HEAD リクエストへのレスポンスのために#####の CloudFront エッジロケーションから供給された合計バイト数。	*region*-Out-Bytes-HTTP-Static: TTL ≥ 3,600 秒のオブジェクトのために HTTP 経由で供給されたバイト数。 *region*-Out-Bytes-HTTPS-Static: TTL ≥ 3,600 秒のオブジェクトのために HTTPS 経由で供給されたバイト数。 *region*-Out-Bytes-HTTP-Dynamic: TTL < 3,600 秒のオブジェクトのために HTTP 経由で提供されたバイト数。 *region*-Out-Bytes-HTTPS-Dynamic: TTL < 3,600 秒のオブジェクトのために HTTPS 経由で配信されたバイト数。 *region*-Out-OBytes-HTTP-Proxy DELETE、OPTIONS、PATCH、POST、および PUT リクエストへの応答として CloudFront からビューワーに HTTP 経由で返されたバイト数。

AWS 請求書の CloudFront 料金	AWS 使用状況レポートの UsageType 列の値
	region-Out-Bytes-HTTPS-Proxy: DELETE、OPTIONS、PATCH、POST、および PUT リクエストへの応答として CloudFront からビューワーに HTTPS 経由で返されたバイト数。
region-DataTransfer-Out-OBytes DELETE、OPTIONS、PATCH、POST、および PUT リクエストへのレスポンスのために CloudFront エッジロケーションからオリジンまたはエッジ関数に転送された合計バイト数。料金には、クライアントからサーバーへの WebSocket データのデータ転送が含まれます。	*region*-Out-OBytes-HTTP-Proxy DELETE、OPTIONS、PATCH、POST、および PUT リクエストへのレスポンスのために CloudFront エッジロケーションからオリジンまたはエッジ関数に HTTP 経由で転送された合計バイト数。 *region*-Out-OBytes-HTTPS-Proxy DELETE、OPTIONS、PATCH、POST、および PUT リクエストへのレスポンスのために CloudFront エッジロケーションからオリジンまたはエッジ関数に HTTPS 経由で転送された合計バイト数。
region-Requests-Tier1 HTTP GET および HEAD リクエストの数。	*region*-Requests-HTTP-Static TTL ≥ 3,600 秒のオブジェクトのために供給された HTTP GET および HEAD リクエストの数。 *region*-Requests-HTTP-Dynamic TTL < 3,600 秒のオブジェクトのために供給された HTTP GET および HEAD リクエストの数。

AWS 請求書の CloudFront 料金	AWS 使用状況レポートの UsageType 列の値
region-Requests-Tier2-HTTPS HTTPS GET および HEAD リクエストの数。	*region*-Requests-HTTPS-Static TTL ≥ 3,600 秒のオブジェクトのために供給された HTTPS GET および HEAD リクエストの数。 *region*-Requests-HTTPS-Dynamic TTL < 3,600 秒のオブジェクトのために供給された HTTPS GET および HEAD リクエストの数。
region-Requests-HTTP-Proxy CloudFront によってオリジンまたはエッジ関数に転送される HTTP DELETE、OPTIONS、PATCH、POST、および PUT リクエストの数 CloudFront がオリジンまたはエッジ関数に転送する HTTP WebSocket リクエスト (Upgrade: websocket ヘッダー付きの GET リクエスト) の数も含まれます。	*region*-Requests-HTTP-Proxy CloudFront 請求書の対応する項目と同じです。
region-Requests-HTTPS-Proxy CloudFront によってオリジンまたはエッジ関数に転送される HTTPS DELETE、OPTIONS、PATCH、POST、および PUT リクエストの数。 CloudFront がオリジンまたはエッジ関数に転送する HTTPS WebSocket リクエスト (Upgrade: websocket ヘッダー付きの GET リクエスト) の数も含まれます。	*region*-Requests-HTTPS-Proxy CloudFront 請求書の対応する項目と同じです。

AWS 請求書の CloudFront 料金	AWS 使用状況レポートの UsageType 列の値
region-Requests-HTTPS-Proxy-FLE CloudFront によってオリジンまたはエッジ関数に転送され、フィールドレベル暗号化を使用して処理された HTTPS DELETE、OPTIONS、PATCH、および POST リクエストの数。	*region*-Requests-HTTPS-Proxy-FLE CloudFront 請求書の対応する項目と同じです。
region-Bytes-OriginShield オリジンから任意のリージョン別エッジキャッシュに転送された合計バイト数。Origin Shield として有効になっているリージョン別エッジキャッシュを含みます。	*region*-Bytes-OriginShield CloudFront 請求書の対応する項目と同じです。
region-OBytes-OriginShield 任意のリージョン別エッジキャッシュからオリジンに転送された合計バイト数。Origin Shield として有効になっているリージョン別エッジキャッシュを含みます。	*region*-OBytes-OriginShield CloudFront 請求書の対応する項目と同じです。
region-Requests-OriginShield 増分レイヤーとして Origin Shield に送信されたリクエストの数。オリジンにプロキシ化される動的 (キャッシュ不可能な) リクエストの場合、Origin Shield は常に増分レイヤーとなります。キャッシュ可能なリクエストの場合、Origin Shield は増分レイヤーになることがあります。 詳細については、「the section called “Origin Shield のコストの見積もり”」を参照してください。	*region*-Requests-OriginShield CloudFront 請求書の対応する項目と同じです。

AWS 請求書の CloudFront 料金	AWS 使用状況レポートの UsageType 列の値
無効化 オブジェクトを無効化する (オブジェクトを CloudFront エッジロケーションから削除する) ための料金。詳細については、「ファイルの無効化に対する支払い」を参照してください。	無効化 CloudFront 請求書の対応する項目と同じです。
SSL-Cert-Custom デフォルトの CloudFront SSL 証明書と CloudFront がお客様のディストリビューションに割り当てたドメイン名を使用する代わりに、example.com などの CloudFront 代替ドメイン名によって SSL 証明書を使用する場合の料金	SSL-Cert-Custom CloudFront 請求書の対応する項目と同じです。
RealTimeLog-KinesisDataStream リアルタイムログに対して生成された行数の料金。	RealTimeLog-KinesisDataStream CloudFront 請求書の対応する項目と同じです。
Executions-CloudFrontFunctions CloudFront Functions の呼び出し数の料金。	Executions-CloudFrontFunctions CloudFront 請求書の対応する項目と同じです。
region-Lambda-Edge-Request Lambda@Edge 関数の呼び出し数の料金。	***region***-Lambda-Edge-Request CloudFront 請求書の対応する項目と同じです。
region-Lambda-Edge-GB-Second Lambda@Edge 関数を呼び出してから、関数が返るか終了するまでの期間の料金。	***region***-Lambda-Edge-GB-Second CloudFront 請求書の対応する項目と同じです。

AWS 請求書の CloudFront 料金	AWS 使用状況レポートの UsageType 列の値
KeyValueStore-EdgeReads CloudFront KeyValueStore メソッド、get()、exists()、meta() への読み取り呼び出し数の料金。詳細については、「キー値ストアのヘルパーメソッド」を参照してください。	KeyValueStore-EdgeReads CloudFront 請求書の対応する項目と同じです。
KeyValueStore-APIOperations CloudFront KeyValueStore API への呼び出し数の料金。	KeyValueStore-APIOperations CloudFront 請求書の対応する項目と同じです。

コンソールで CloudFront レポートを表示する

CloudFront アクティビティの次のレポートをコンソールで表示できます。

トピック

- CloudFront キャッシュ統計レポートを表示する
- CloudFront 人気オブジェクトのレポートを表示する
- CloudFront トップリファラーレポートを表示する
- CloudFront 使用状況レポートを表示する
- CloudFront ビューワーレポートを表示する

これらのレポートのほとんどは、CloudFront アクセスログのデータに基づいており、このログには CloudFront が受け取ったすべてのユーザーリクエストの詳細が含まれています。このレポートを表示するために、アクセスログを有効にする必要はありません。詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

CloudFront キャッシュ統計レポートを表示する

Amazon CloudFront キャッシュ統計レポートには次の情報が含まれます。

- Total requests – すべての HTTP ステータスコード (200、404 など) およびすべてのメソッド (GET、HEAD、POST など) に対するリクエストの総数を示します。

- Percentage of viewer requests by result type – 選択された CloudFront ディストリビューションのビューアリクエスト合計からヒット、ミス、エラーの割合を示します。
- Bytes transferred to viewers – バイト総数とミスのバイト数を示します。
- HTTP status codes – HTTP ステータスコード別のビューワーリクエスト数を示します。
- Percentage of GET requests that didn't finish downloading – リクエストされたオブジェクトのダウンロードを終了しなかったビューワーの GET リクエストが、リクエスト総数に占める割合 (%) を示します。

これらの統計のデータは CloudFront アクセスログと同じソースから取得されますが、キャッシュ統計を表示するためにアクセスログを有効にする必要はありません。

毎時間または毎日のデータポイントを使用して、過去 60 日間の指定した日付範囲のグラフを表示できます。通常は、1 時間前までに CloudFront が受け取ったリクエストについてデータを表示できますが、データが 24 時間ほど遅れることもあります。

トピック

- コンソールで CloudFront キャッシュ統計レポートを表示する
- CSV 形式でデータをダウンロードする
- キャッシュ統計のグラフと CloudFront 標準ログ (アクセスログ) のデータとの関連

コンソールで CloudFront キャッシュ統計レポートを表示する

コンソールで CloudFront キャッシュ統計レポートを表示できます。

CloudFront キャッシュ統計を表示するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[キャッシュ統計] をクリックします。
3. [CloudFront Cache Statistics Reports (CloudFront キャッシュ統計レポート)] ペインの [Start Date (開始日)] と [End Date (終了日)] で、キャッシュ統計のグラフを表示する日付範囲を選択します。使用できる範囲は、[Granularity] で選択した値によって決まります。

    - Daily – 1 日につき 1 つのデータポイントを使用してグラフを表示するには、過去 60 日の中で任意の日付範囲を選択します。

- Hourly – 1 時間につき 1 つのデータポイントを使用してグラフを表示するには、過去 60 日以内で最大 14 日間の任意の日付範囲を選択します。

日付と時刻は協定世界時 (UTC) です。

4. [Granularity] では、グラフに 1 日につき 1 つのデータポイントを表示するか、1 時間につき 1 つのデータポイントを表示するかを指定します。14 日を超える日付範囲を指定した場合、1 時間につき 1 つのデータポイントを指定することはできなくなります。
5. [Viewer Location] で、ビューワーのリクエストが発信された大陸を選択するか、[All Locations] を選択します。キャッシュ統計のグラフには、指定した場所から CloudFront が受信したリクエストのデータが含まれます。
6. [Distribution] リストでは、使用状況グラフにデータを表示するディストリビューションを選択します。

   - 個々のディストリビューション - 選択した CloudFront ディストリビューションのデータがグラフに表示されます。[Distribution] リストには、ディストリビューションのディストリビューション ID と代替ドメイン名 (CNAME) が表示されます (ある場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。
   - All distributions - 現在の AWS アカウントに関連付けられているすべてのディストリビューションのデータが集計されてグラフに表示されます。ただし、削除したディストリビューションは除外されます。
7. [Update] (更新) を選択します。

グラフ内の毎日または毎時間のデータポイントのデータを表示するには、データポイントの上にカーソルを合わせます。

転送データを示すグラフの場合、各グラフの Y 軸の単位をギガバイト、メガバイト、キロバイトのいずれかに変更できることに注意してください。

CSV 形式でデータをダウンロードする

キャッシュ統計レポートは CSV 形式でダウンロードできます。このセクションでは、レポートをダウンロードする方法と、レポートの値について説明します。

キャッシュ統計レポートを CSV 形式でダウンロードするには

1. キャッシュ統計レポートを表示しているときに、[CSV] を選択します。

2. [Opening file name] ダイアログボックスで、ファイルを開くか保存するかを選択します。

レポートに関する情報

レポートの先頭数行には次の情報が含まれます。

バージョン

この CSV ファイルの形式のバージョン。

レポート

レポートの名前。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

StartDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の開始日。

EndDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の終了日。

GeneratedTimeUTC

協定世界時 (UTC) によるレポートを実行した日時。

詳細度

レポートの各行が 1 時間と 1 日のどちらを表すか。

ViewerLocation

ビューワーリクエストが発信された大陸。または、すべての場所についてレポートをダウンロードする場合は ALL。

キャッシュ統計レポートのデータ

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

ViewerLocation

ビューワーリクエストが発信された大陸。または、すべての場所についてレポートをダウンロードする場合は ALL。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

RequestCount

すべての HTTP ステータスコード (200、404 など) およびすべてのメソッド (GET、HEAD、POST など) のリクエストの総数。

HitCount

CloudFront エッジキャッシュからオブジェクトが提供されたビューワーリクエストの数。

MissCount

オブジェクトが現在エッジキャッシュに存在せず、CloudFront でオリジンからオブジェクトを取得する必要があるビューワーリクエストの数。

ErrorCount

エラーになり、CloudFront でオブジェクトを提供できなかったビューワーリクエストの数。

IncompleteDownloadCount

ビューワーがオブジェクトのダウンロードを開始したが、ダウンロードを終了できなかったビューワーリクエストの数。

HTTP2xx

HTTP ステータスコードが 2xx 値 (成功) であるビューワーリクエストの数。

HTTP3xx

HTTP ステータスコードが 3xx 値 (追加のアクションが必要) であるビューワーリクエストの数。

HTTP4xx

HTTP ステータスコードが 4xx 値 (クライアントエラー) であるビューワーリクエストの数。

HTTP5xx

HTTP ステータスコードが 5xx 値 (サーバーエラー) であるビューワーリクエストの数。

TotalBytes

すべての HTTP メソッドに対するすべてのリクエストに応じて CloudFront からビューワーに提供される合計バイト数。

BytesFromMisses

リクエストの発生時にエッジキャッシュに存在しなかったオブジェクトのビューワーに提供されたバイト数。この値は、オリジンから CloudFront エッジキャッシュに転送されたバイトの正確な概算です。ただし、エッジキャッシュに既に存在していても、有効期限が切れているオブジェクトのリクエストは除きます。

キャッシュ統計のグラフと CloudFront 標準ログ (アクセスログ) のデータとの関連

次の表は、CloudFront コンソールのキャッシュ統計のグラフと、対応する CloudFront アクセスログの値を示します。CloudFront アクセスログの詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

Total requests

このグラフには、すべての HTTP ステータスコード (200 または 404 など) およびすべてのメソッド (GET、HEAD、または POST など) のリクエストの総数が表示されます。このグラフに表示されるリクエストの総数は、同じ期間のアクセスログファイルのリクエストの総数と同じです。

Percentage of viewer requests by result type

このグラフには、選択した CloudFront ディストリビューションの合計ビューワーリクエストに対するヒット、ミス、エラーの割合が表示されます。

- Hit – オブジェクトが CloudFront エッジキャッシュから提供されるビューワーリクエスト。アクセスログでは、これらのリクエストの `x-edge-response-result-type` の値は `Hit` です。
- Miss – オブジェクトが現在エッジキャッシュに存在せず、CloudFront でオリジンからオブジェクトを取得する必要があるビューワーリクエスト。アクセスログでは、これらのリクエストの `x-edge-response-result-type` の値は `Miss` です。

- Error – エラーになり、CloudFront でオブジェクトを提供できなかったビューワーリクエスト。アクセスログでは、これらのリクエストの `x-edge-response-result-type` の値は `Error`、`LimitExceeded` または `CapacityExceeded` です。

グラフには、エッジキャッシュに存在しても、有効期限が切れているオブジェクトのリフレッシュヒットリクエストは含まれません。アクセスログでは、リフレッシュヒットのリクエストの `x-edge-response-result-type` の値は `RefreshHit` です。

Bytes transferred to viewers

このグラフには 2 つの値が表示されます。

- Total bytes – すべての HTTP メソッドに対するすべてのリクエストに応じて CloudFront からビューワーに提供される合計バイト数。CloudFront アクセスログでは、[Total Bytes (合計バイト数)] は、`sc-bytes` 列の同じ期間に発生したすべてのリクエストの値の合計です。
- Bytes from misses – リクエストの発生時にエッジキャッシュに存在しなかったオブジェクトのビューワーに提供されたバイト数。CloudFront アクセスログでは、[bytes from misses]] (欠落しているオブジェクトのバイト数) は、`sc-bytes` 列で、`x-edge-result-type` の値が `Miss` であるリクエストの値の合計です。この値は、オリジンから CloudFront エッジキャッシュに転送されたバイトの正確な概算です。ただし、エッジキャッシュに既に存在していても、有効期限が切れているオブジェクトのリクエストは除きます。

HTTP ステータスコード

このグラフには HTTP ステータスコードごとのビューワーリクエストが表示されます。CloudFront アクセスログでは、ステータスコードは `sc-status` 列に表示されます。

- 2xx – 成功したリクエスト。
- 3xx – 追加のアクションが必要です。たとえば、301 (Moved Permanently) は、リクエストされたオブジェクトが異なる場所に移動されていることを意味します。
- 4xx – クライアント側のエラー。たとえば、404 (Not Found) は、クライアントが、検出できないオブジェクトをリクエストしたことを意味します。
- 5xx – オリジンサーバーがリクエストを実行しませんでした。たとえば、503 (Service Unavailable) は、オリジンサーバーが現在利用できないことを意味します。

Percentage of GET requests that didn't finish downloading

このグラフでは、合計リクエストに対して、リクエストされたオブジェクトのダウンロードが終了していない、ビューワーの `GET` リクエストの割合が表示されます。通常、オブジェクトのダウンロードが完了しないのは、たとえば別のリンクをクリックしたり、ブラウザを閉じたりして、

ビューワーによってキャンセルされたときです。CloudFront アクセスログでは、これらのリクエストの 200 列の値は、sc-status で、Error 列の値は、x-edge-result-type です。

CloudFront 人気オブジェクトのレポートを表示する

Amazon CloudFront 人気オブジェクトレポートを表示して、過去 60 日間の指定された日付範囲内のディストリビューションで最も人気のある 50 個のオブジェクトを確認します。これらのオブジェクトに関する統計を表示することもできます。これには、以下が含まれます。

- オブジェクトに対するリクエストの数
- ヒットとミスの数
- Hit ratio
- ミスのために供給されたバイト数
- 供給された合計バイト数
- 不完全なダウンロードの数
- HTTP ステータスコード (2xx、3xx、4xx、5xx) 別のリクエスト数

これらの統計のデータは CloudFront アクセスログと同じソースから取得されますが、人気オブジェクトを表示するためにアクセスログを有効にする必要はありません。

トピック

コンソールで CloudFront 人気オブジェクトのレポートを表示する

コンソールで CloudFront 人気オブジェクトのレポートを表示できます。

CloudFront ディストリビューションの人気オブジェクトを表示するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[人気オブジェクト] をクリックします。

3. [CloudFront Popular Objects Report (CloudFront 人気オブジェクトレポート)] ペインの [Start Date (開始日)] と [End Date (終了日)] で、人気オブジェクトのリストを表示する日付範囲を選択します。過去 60 日間の任意の日付範囲を選択できます。

   日付と時刻は協定世界時 (UTC) です。
4. [Distribution] リストで、人気オブジェクトのリストを表示するディストリビューションを選択します。
5. [更新] を選択します。

CloudFront が人気オブジェクト統計を計算する方法

ディストリビューションの上位 50 件のオブジェクトの正確な数を取得するには、CloudFront で、午前 0 時から 10 分間隔ですべてのオブジェクトのリクエストをカウントし、その後 24 時間にわたって上位 150 件のオブジェクトの現在までの累計を保持します。(CloudFront は、上位 150 件のオブジェクトの毎日の合計を 60 日間保持します)。

リストの最下位に近いオブジェクトは、リストに加わったり、リストからなくなったりするため、これらのオブジェクトの合計は概算です。150 件のオブジェクトのリストの中の上位 50 件のオブジェクトもリスト内で上がったり下がったりする可能性はありますが、リストから完全になくなることはほとんどないため、これらのオブジェクトの合計は信頼できます。

オブジェクトが上位 150 件のリストからなくなり、その日のうちに再びリストに加わった場合、CloudFront で、そのオブジェクトがリストになかった期間の推定リクエスト数が追加されます。この予測は、その期間中にリストの最後にあったオブジェクトから受け取ったリクエストの数に基づいています。

オブジェクトがその日のうちに上位 50 件のオブジェクトに加わった場合、オブジェクトが上位 150 件になかった間に CloudFront が受け取ったリクエストの数の予想により、通常、人気オブジェクトレポートのリクエスト数は、そのオブジェクトのアクセスログに表示されるリクエストの数を超えます。

CSV 形式でデータをダウンロードする

人気オブジェクトレポートは CSV 形式でダウンロードできます。このセクションでは、レポートをダウンロードする方法と、レポートの値について説明します。

人気オブジェクトレポートを CSV 形式でダウンロードするには

1. 人気オブジェクトレポートを表示しているときに、[CSV] を選択します。

2. [Opening file name] ダイアログボックスで、ファイルを開くか保存するかを選択します。

レポートに関する情報

レポートの先頭数行には次の情報が含まれます。

バージョン

この CSV ファイルの形式のバージョン。

レポート

レポートの名前。

DistributionID

レポートを実行した対象のディストリビューションの ID。

StartDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の開始日。

EndDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の終了日。

GeneratedTimeUTC

協定世界時 (UTC) によるレポートを実行した日時。

人気オブジェクトレポートのデータ

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

オブジェクト

オブジェクトの URL の最後の 500 文字。

RequestCount

このオブジェクトに対するリクエストの総数。

HitCount

CloudFront エッジキャッシュからオブジェクトが提供されたビューワーリクエストの数。

MissCount

オブジェクトが現在エッジキャッシュに存在せず、CloudFront でオリジンからオブジェクトを取得する必要があるビューワーリクエストの数。

HitCountPct

`HitCount` の値に対する `RequestCount` の値の割合。

BytesFromMisses

リクエストの発生時にエッジキャッシュにオブジェクトが存在しなかった場合に、このオブジェクトについてビューワーに提供されたバイト数。

TotalBytes

すべての HTTP メソッドのすべてのリクエストに応じて、このオブジェクトについて CloudFront からビューワーに提供される合計バイト数。

IncompleteDownloadCount

このオブジェクトについて、ビューワーがオブジェクトのダウンロードを開始したが、ダウンロードを終了できなかったビューワーリクエストの数。

HTTP2xx

HTTP ステータスコードが 2xx 値 (成功) であるビューワーリクエストの数。

HTTP3xx

HTTP ステータスコードが 3xx 値 (追加のアクションが必要) であるビューワーリクエストの数。

HTTP4xx

HTTP ステータスコードが 4xx 値 (クライアントエラー) であるビューワーリクエストの数。

HTTP5xx

HTTP ステータスコードが 5xx 値 (サーバーエラー) であるビューワーリクエストの数。

人気オブジェクトレポートのデータと CloudFront 標準ログ (アクセスログ) のデータとの関連

次のリストは、CloudFront コンソールの人気オブジェクトレポートの値と、対応する CloudFront アクセスログの値を示します。CloudFront アクセスログの詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

URL

オブジェクトへのアクセスにビューワーが使用する URL の末尾 500 文字です。

リクエスト

オブジェクトに対するリクエストの総数。一般的にこの値は、CloudFront アクセスログのオブジェクトの `GET` リクエストの数とほぼ一致します。

Hits

CloudFront エッジキャッシュからオブジェクトが提供されたビューワーリクエストの数。アクセスログでは、これらのリクエストの `x-edge-response-result-type` の値は `Hit` です。

Misses

オブジェクトがエッジキャッシュに存在しないため、CloudFront でオリジンからオブジェクトが取得されたビューワーリクエストの数。アクセスログでは、これらのリクエストの `x-edge-response-result-type` の値は `Miss` です。

Hit ratio

[Requests] 列の値に対する、[Hits] 列の値の割合。

Bytes from misses

リクエストの発生時にエッジキャッシュに存在しなかったオブジェクトのビューワーに提供されたバイト数。CloudFront アクセスログでは、[bytes from misses]] (欠落しているオブジェクトのバイト数) は、`sc-bytes` 列で、`x-edge-result-type` の値が `Miss` であるリクエストの値の合計です。

Total bytes

すべての HTTP メソッドのオブジェクトに対するすべてのリクエストに応じて CloudFront からビューワーに提供される合計バイト数。CloudFront アクセスログでは、[total bytes] (合計バイト数) は、`sc-bytes` 列の同じ期間に発生したすべてのリクエストの値の合計です。

Incomplete downloads

リクエストされたオブジェクトのダウンロードが終了しなかったビューワーリクエストの数。通常、ダウンロードが完了しないのは、たとえば別のリンクをクリックしたり、ブラウザを閉じたりして、ビューワーによってキャンセルされたときです。CloudFront アクセスログでは、これらのリクエストの `200` 列の値は、`sc-status` で、`Error` 列の値は、`x-edge-result-type` です。

2xx

HTTP ステータスコードが `2xx`、`Successful` であるリクエストの数。CloudFront アクセスログでは、ステータスコードは `sc-status` 列に表示されます。

3xx

HTTP ステータスコードが `3xx` (`Redirection`) であるリクエストの数です。`3xx` のステータスコードは追加のアクションが必要であることを表します。たとえば、301 (Moved Permanently) は、リクエストされたオブジェクトが異なる場所に移動されていることを意味します。

4xx

HTTP ステータスコードが `4xx` (`Client Error`) であるリクエストの数です。`4xx` のステータスコードはクライアント側でエラーが発生したことを表します。たとえば、404 (Not Found) は、クライアントが、検出できないオブジェクトをリクエストしたことを意味します。

5xx

HTTP ステータスコードが `5xx` (`Server Error`) であるリクエストの数です。`5xx` のステータスコードはオリジンサーバーでリクエストが実行されなかったことを表します。たとえば、503 (Service Unavailable) は、オリジンサーバーが現在利用できないことを意味します。

CloudFront トップリファラーレポートを表示する

CloudFront トップリファラーレポートには、過去 60 日間の任意の日付範囲について、次の項目が含まれます。

- 上位 25 のリファラー (CloudFront がディストリビューションに配信しているオブジェクトに対する HTTP および HTTPS リクエストが最も多いウェブサイトのドメイン)
- リファラーからのリクエストの数
- 指定した期間のリクエストの総数に対してリファラからのリクエストの数の割合。

トップリファラーレポートのデータは CloudFront アクセスログと同じソースから取得されますが、トップリファラーを表示するためにアクセスログを有効にする必要はありません。

トップリファラーは、検索エンジン、オブジェクトに直接リンクされた他のウェブサイト、またはユーザー自身のウェブサイトである場合もあります。例えば、`https://example.com/index.html` が 10 個のグラフィックにリンクする場合、`example.com` は 10 個のグラフィックすべてのリファラです。

> **Note**
>
> ユーザーがブラウザのアドレス行に直接 URL を入力した場合、リクエストしたオブジェクトのリファラはありません。

トピック

コンソールで CloudFront トップリファラーレポートを表示する

CloudFront トップリファラーレポートはコンソールで表示できます。

CloudFront ディストリビューションのトップリファラーを表示するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[トップリファラー] を選択します。
3. [CloudFront Top Referrers Report (CloudFront トップリファラレポート)] ペインの [Start Date (開始日)] と [End Date (終了日)] で、トップリファラのリストを表示する日付範囲を選択します。

   日付と時刻は協定世界時 (UTC) です。
4. [Distribution] リストで、トップリファラのリストを表示するディストリビューションを選択します。

5. [Update] (更新) を選択します。

CloudFront がトップリファラー統計を計算する方法

上位 25 件のリファラの正確な数を取得するには、CloudFront ですべてのオブジェクトのリクエストを 10 分間隔でカウントし、上位 75 件のリファラの現在までの累計を保持します。リストの最下位に近いリファラーは、リストに加わったり、リストからなくなったりするため、これらのリファラーの合計は概算です。

75 件のリファラーのリストの中の上位 25 件のリファラーもリスト内で上がったり下がったりする可能性はありますが、リストから完全になくなることはほとんどないため、通常これらのリファラーの合計は信頼できます。

CSV 形式でデータをダウンロードする

トップリファラレポートは CSV 形式でダウンロードできます。このセクションでは、レポートをダウンロードする方法と、レポートの値について説明します。

トップリファラレポートを CSV 形式でダウンロードするには

1. トップリファラーレポートを表示しているときに、[CSV] を選択します。
2. [Opening file name] ダイアログボックスで、ファイルを開くか保存するかを選択します。

レポートに関する情報

レポートの先頭数行には次の情報が含まれます。

バージョン

この CSV ファイルの形式のバージョン。

レポート

レポートの名前。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

StartDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の開始日。

EndDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の終了日。

GeneratedTimeUTC

協定世界時 (UTC) によるレポートを実行した日時。

トップリファラレポートのデータ

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

Referrer

リファラのドメイン名。

RequestCount

[Referrer] 列のドメイン名からのリクエストの総数。

RequestsPct

指定した期間のリクエストの総数に対してリファラによって送信されたリクエストの数の割合。

トップリファラレポートのデータと CloudFront 標準ログ (アクセスログ) のデータとの関連

次のリストは、CloudFront コンソールのトップリファラレポートの値と、対応する CloudFront アクセスログの値を示します。CloudFront アクセスログの詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

Referrer

リファラのドメイン名。アクセスログでは、リファラは cs(Referer) 列に表示されます。

Request count

[Referrer] 列のドメイン名からのリクエストの総数。一般的にこの値は、CloudFront アクセスログのリファラからの GET リクエストの数とほぼ一致します。

リクエスト %

指定した期間のリクエストの総数に対してリファラによって送信されたリクエストの数の割合。リファラが 25 個以上あると、[request count] (リクエスト数) 列に指定した期間のすべてのリクエストを含めることができないため、このテーブルのデータに基づいて [Request %] (リクエスト %) を計算することはできません。

CloudFront 使用状況レポートを表示する

CloudFront 使用状況レポートには次の情報が含まれます。

- Number of requests – 指定された CloudFront ディストリビューションの時間間隔ごとに、選択されたリージョンのエッジロケーションから CloudFront が応答したリクエストの総数を示します。
- Data transferred by protocol および data transferred by destination – どちらの場合も、指定された CloudFront ディストリビューションの時間間隔ごとに、選択されたリージョンの CloudFront エッジロケーションから転送されたデータの合計量を示します。データは、以下のように異なる方法で分けられます。
  - プロトコル別 – プロトコル (HTTP または HTTPS) 別にデータを分けます。
  - 送信先別 – 送信先 (ビューワーまたはオリジン) 別にデータを分けます。

CloudFront 使用状況レポートは、特別な設定を必要としない、CloudFront 用の AWS 使用状況レポートに基づいています。詳細については、「CloudFront の AWS 使用状況レポートを表示する」を参照してください。

毎時間または毎日のデータポイントを使用して、過去 60 日間の指定した日付範囲のレポートを表示できます。通常、最近 4 時間前までに CloudFront が受け取ったリクエストについてデータを表示できますが、ときにはデータが 24 時間、遅れることがあります。

詳細については、「使用状況グラフと CloudFront 使用状況レポートのデータとの関連」を参照してください。

トピック

- コンソールで CloudFront 使用状況レポートを表示する

- CSV 形式でデータをダウンロードする
- 使用状況グラフと CloudFront 使用状況レポートのデータとの関連

コンソールで CloudFront 使用状況レポートを表示する

コンソールで CloudFront 使用状況レポートを表示できます。

CloudFront 使用状況レポートを表示するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで [使用状況レポート] を選択します。
3. [CloudFront Usage Reports (CloudFront 使用状況レポート)] ペインの [Start Date (開始日)] と [End Date (終了日)] で、使用状況グラフを表示する日付範囲を選択します。使用できる範囲は、[Granularity] で選択した値によって決まります。

   - Daily — 1 日につき 1 つのデータポイントを使用してグラフを表示するには、過去 60 日の中で任意の日付範囲を選択します。
   - Hourly — 1 時間につき 1 つのデータポイントを使用してグラフを表示するには、過去 60 日以内で最大 14 日間の任意の日付範囲を選択します。

   日付と時刻は協定世界時 (UTC) です。
4. [Granularity] では、グラフに 1 日につき 1 つのデータポイントを表示するか、1 時間につき 1 つのデータポイントを表示するかを指定します。14 日を超える日付範囲を指定した場合、1 時間につき 1 つのデータポイントを指定することはできなくなります。
5. [Billing Region (請求リージョン)] では、データを表示する CloudFront 請求リージョンを選択するか、[All Regions (すべてのリージョン)] を選択します。使用状況グラフは、指定したリージョンのエッジロケーションで CloudFront が処理するリクエストのデータを含みます。CloudFront がリクエストを処理するリージョンは、ビューワーの場所に対応している場合も、対応していない場合もあります。

   ディストリビューションの料金クラスに含まれるリージョンのみを選択します。そうしないと、使用状況グラフにデータが含まれない可能性があります。たとえば、ディストリビューションで価格クラス 200 を選択した場合、南米およびオーストラリアの請求リージョンは含まれません。そのため、CloudFront は一般にこれらのリージョンからのリクエストを処理しません。料金クラスの詳細については、「CloudFront の料金」を参照してください。

6. [Distribution] リストでは、使用状況グラフにデータを表示するディストリビューションを選択します。

   - 個々のディストリビューション - 選択した CloudFront ディストリビューションのデータがグラフに表示されます。[Distribution] リストには、ディストリビューションのディストリビューション ID と代替ドメイン名 (CNAME) が表示されます (ある場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。
   - All distributions (excludes deleted) - 現在の AWS アカウントに関連付けられているすべてのディストリビューションのデータが集計されてグラフに表示されます。ただし、削除したディストリビューションは除外されます。
   - All Deleted Distributions - 現在の AWS アカウントに関連付けられていて過去 60 日間に削除されたすべてのディストリビューションのデータが集計されてグラフに表示されます。
7. [グラフを更新] を選択します。

グラフ内の毎日または毎時間のデータポイントのデータを表示するには、データポイントの上にカーソルを合わせます。

転送データを示すグラフの場合、各グラフの Y 軸の単位をギガバイト、メガバイト、キロバイトのいずれかに変更できることに注意してください。

CSV 形式でデータをダウンロードする

使用状況レポートは CSV 形式でダウンロードできます。このセクションでは、レポートをダウンロードする方法と、レポートの値について説明します。

使用状況レポートを CSV 形式でダウンロードするには

1. 使用状況レポートを表示しているときに、[CSV] を選択します。
2. [Opening file name] ダイアログボックスで、ファイルを開くか保存するかを選択します。

レポートに関する情報

レポートの先頭数行には次の情報が含まれます。

バージョン

この CSV ファイルの形式のバージョン。

レポート

レポートの名前。

DistributionID

レポートを実行した対象のディストリビューションの ID。すべてのディストリビューションを対象にレポートを実行した場合は ALL。削除したディストリビューションを対象にレポートを実行した場合は ALL_DELETED。

StartDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の開始日。

EndDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の終了日。

GeneratedTimeUTC

協定世界時 (UTC) によるレポートを実行した日時。

詳細度

レポートの各行が 1 時間と 1 日のどちらを表すか。

BillingRegion

ビューワーリクエストが発信された大陸。または、すべての請求リージョンについてレポートをダウンロードする場合は ALL。

使用状況レポートのデータ

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。すべてのディストリビューションを対象にレポートを実行した場合は ALL。削除したディストリビューションを対象にレポートを実行した場合は ALL_DELETED。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

BillingRegion

レポートを実行した対象の CloudFront 請求リージョン、または ALL。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

HTTP

指定した CloudFront ディストリビューションで時間間隔ごとに、CloudFront が応答する、選択したリージョンのエッジロケーションからの HTTP リクエストの数。値には以下のものが含まれます。

- CloudFront からビューワーへのデータ転送を引き起こす GET リクエストと HEAD リクエストの数
- CloudFront がオリジンにデータを転送する、DELETE、OPTIONS、PATCH、POST、および PUT リクエストの数

HTTPS

指定した CloudFront ディストリビューションで時間間隔ごとに、CloudFront が応答する、選択したリージョンのエッジロケーションからの HTTPS リクエストの数。値には以下のものが含まれます。

- CloudFront からビューワーへのデータ転送を引き起こす GET リクエストと HEAD リクエストの数
- CloudFront がオリジンにデータを転送する、DELETE、OPTIONS、PATCH、POST、および PUT リクエストの数

HTTPBytes

指定した CloudFront ディストリビューションで期間中に、選択した請求リージョンの CloudFront エッジロケーションから HTTP 経由で転送されるデータの合計量。値には以下のものが含まれます。

- GET リクエストと HEAD リクエストに応じて CloudFront からビューワーに転送されるデータ
- DELETE、OPTIONS、PATCH、POST、PUT の各リクエストに応じて、ビューワーから CloudFront に転送されるデータ
- DELETE、OPTIONS、PATCH、POST、PUT の各リクエストに応じて CloudFront からビューワーに転送されるデータ

HTTPSBytes

指定した CloudFront ディストリビューションで期間中に、選択した請求リージョンの CloudFront エッジロケーションから HTTPS 経由で転送されるデータの合計量。値には以下のものが含まれます。

- GET リクエストと HEAD リクエストに応じて CloudFront からビューワーに転送されるデータ
- DELETE、OPTIONS、PATCH、POST、PUT の各リクエストに応じて、ビューワーから CloudFront に転送されるデータ
- DELETE、OPTIONS、PATCH、POST、PUT の各リクエストに応じて CloudFront からビューワーに転送されるデータ

BytesIn

指定した CloudFront ディストリビューションで時間間隔ごとに、選択したリージョンの DELETE、OPTIONS、PATCH、POST、および PUT リクエストについて、CloudFront からオリジンに転送されるデータの合計量。

BytesOut

指定した CloudFront ディストリビューションにおいて時間間隔ごとに、選択したリージョンで CloudFront からビューワーに HTTP および HTTPS 経由で転送されるデータの合計量。値には以下のものが含まれます。

- GET リクエストと HEAD リクエストに応じて CloudFront からビューワーに転送されるデータ
- DELETE、OPTIONS、PATCH、POST、PUT の各リクエストに応じて CloudFront からビューワーに転送されるデータ

使用状況グラフと CloudFront 使用状況レポートのデータとの関連

次のリストは、CloudFront コンソールの使用状況グラフと、対応する CloudFront 使用状況レポートの [Usage Type (使用タイプ)] 列の値を示します。

トピック

リクエストの数

このグラフは、指定された CloudFront ディストリビューションの時間間隔ごとに、選択されたリージョンのエッジロケーションから CloudFront が応答したリクエストの総数を、プロトコル別 (HTTP または HTTPS) およびタイプ別 (静的、動的、またはプロキシ) に分けて示します。

Number of HTTP requests

- *region*-Requests-HTTP-Static: TTL ≥ 3600 秒のオブジェクトのために供給された HTTP GET および HEAD リクエストの数
- *region*-Requests-HTTP-Dynamic: TTL < 3600 秒のオブジェクトのために供給された HTTP GET および HEAD リクエストの数
- *region*-Requests-HTTP-Proxy: CloudFront がお客様のオリジンに転送した HTTP DELETE、OPTIONS、PATCH、POST、PUT リクエストの数

Number of HTTPS requests

- *region*-Requests-HTTPS-Static: TTL ≥ 3600 秒のオブジェクトのために供給された HTTPS GET および HEAD リクエストの数
- *region*-Requests-HTTPS-Dynamic: TTL < 3600 秒のオブジェクトのために供給された HTTPS GET および HEAD リクエストの数
- *region*-Requests-HTTPS-Proxy: CloudFront がお客様のオリジンに転送した HTTPS DELETE、OPTIONS、PATCH、POST、PUT リクエストの数

プロトコルごとのデータ転送

このグラフは、指定した CloudFront ディストリビューションにおいて時間間隔ごとに、選択したリージョンの CloudFront エッジロケーションから転送されたデータの合計量を、プロトコル別 (HTTP または HTTPS)、タイプ別 (静的、動的、またはプロキシ)、および送信先別 (ビューワーまたはオリジン) に分けて示します。

Data transferred over HTTP

- *region*-Out-Bytes-HTTP-Static: TTL >= 3600 秒のオブジェクトのために HTTP 経由で供給されたバイト数
- *region*-Out-Bytes-HTTP-Dynamic: TTL < 3600 秒のオブジェクトのために HTTP 経由で供給されたバイト数
- *region*-Out-Bytes-HTTP-Proxy: DELETE、OPTIONS、PATCH、POST、PUT リクエストへの応答として CloudFront からビューワーに HTTP 経由で返されたバイト数

- *region*-Out-OBytes-HTTP-Proxy: DELETE、OPTIONS、PATCH、POST、PUT リクエストへの応答として CloudFront エッジロケーションからオリジンに HTTP 経由で転送された合計バイト数

Data transferred over HTTPS

- *region*-Out-Bytes-HTTPS-Static: TTL >= 3600 秒のオブジェクトのために HTTPS 経由で供給されたバイト数
- *region*-Out-Bytes-HTTPS-Dynamic: TTL < 3600 秒のオブジェクトのために HTTPS 経由で供給されたバイト数
- *region*-Out-Bytes-HTTPS-Proxy: DELETE、OPTIONS、PATCH、POST、PUT リクエストへの応答として CloudFront からビューワーに HTTPS 経由で返されたバイト数
- *region*-Out-OBytes-HTTPS-Proxy: DELETE、OPTIONS、PATCH、POST、PUT リクエストへの応答として CloudFront エッジロケーションからオリジンに HTTPS 経由で転送された合計バイト数

ディストリビューションごとのデータ転送

このグラフは、指定した CloudFront ディストリビューションにおいて時間間隔ごとに、選択したリージョンの CloudFront エッジロケーションから転送されたデータの合計量を、プロトコル別 (HTTP または HTTPS)、タイプ別 (静的、動的、またはプロキシ)、および送信先別 (ユーザーまたはオリジン) に分けて示します。

CloudFront からビューワーに転送されたデータ

- *region*-Out-Bytes-HTTP-Static: TTL >= 3600 秒のオブジェクトのために HTTP 経由で供給されたバイト数
- *region*-Out-Bytes-HTTPS-Static: TTL >= 3600 秒のオブジェクトのために HTTPS 経由で供給されたバイト数
- *region*-Out-Bytes-HTTP-Dynamic: TTL < 3600 秒のオブジェクトのために HTTP 経由で供給されたバイト数
- *region*-Out-Bytes-HTTPS-Dynamic: TTL < 3600 秒のオブジェクトのために HTTPS 経由で供給されたバイト数
- *region*-Out-Bytes-HTTP-Proxy: DELETE、OPTIONS、PATCH、POST、PUT リクエストへの応答として CloudFront からビューワーに HTTP 経由で返されたバイト数
- *region*-Out-Bytes-HTTPS-Proxy: DELETE、OPTIONS、PATCH、POST、PUT リクエストへの応答として CloudFront からビューワーに HTTPS 経由で返されたバイト数

CloudFront からオリジンに転送されるデータ

- *region*-Out-OBytes-HTTP-Proxy: DELETE、OPTIONS、PATCH、POST、PUT リクエストへの応答として CloudFront エッジロケーションからオリジンに HTTP 経由で転送された合計バイト数
- *region*-Out-OBytes-HTTPS-Proxy: DELETE、OPTIONS、PATCH、POST、PUT リクエストへの応答として CloudFront エッジロケーションからオリジンに HTTPS 経由で転送された合計バイト数

CloudFront ビューワーレポートを表示する

CloudFront ビューワーレポートには、過去 60 日間の任意の日付範囲について、次の情報が含まれます。

- デバイス – コンテンツへのアクセスに最も頻繁に使用されるデバイスのタイプ (デスクトップやモバイルなど)
- ブラウザ – コンテンツへのアクセスに最も頻繁に使用される上位 10 のブラウザ (Chrome や Firefox など)
- オペレーティングシステム – コンテンツへのアクセスに最も頻繁に使用される上位 10 のオペレーティングシステム (Linux、macOS、Windows など)
- ロケーション – コンテンツに最も頻繁にアクセスするビューワーの上位 50 のロケーション (国、または米国の州/準州)。
  - また、過去 60 日間の最大 14 日間の日付範囲で時間単位のデータポイントがあるロケーションを表示することもできます。

ビューワーのグラフおよびレポートを表示するために、アクセスログを有効にする必要はありません。

トピック

- コンソールでビューワーのグラフとレポートを表示する
- CSV 形式でデータをダウンロードする
- ビューワーレポートに含まれるデータ
- ロケーションレポートのデータと CloudFront 標準ログ (アクセスログ) のデータとの関連

コンソールでビューワーのグラフとレポートを表示する

CloudFront ビューワーのグラフとレポートは、コンソールで表示できます。

CloudFront ビューワーのグラフおよびレポートを表示するには

1. AWS Management Consoleにサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. ナビゲーションペインで、[ビューワー] を選択します。
3. [CloudFront Viewers (CloudFront ビューワー)] ペインの [Start Date (開始日)] と [End Date (終了日)] で、ビューワーのグラフおよびレポートを表示する日付範囲を選択します。

   ロケーショングラフで使用できる範囲は、[Granularity] で選択した値によって異なります。

   - Daily – 1 日につき 1 つのデータポイントを使用してグラフを表示するには、過去 60 日の中で任意の日付範囲を選択します。
   - Hourly – 1 時間につき 1 つのデータポイントを使用してグラフを表示するには、過去 60 日以内で最大 14 日間の任意の日付範囲を選択します。

   日付と時刻は協定世界時 (UTC) です。
4. (ブラウザとオペレーティングシステムのグラフのみ) [Grouping] で、ブラウザおよびオペレーティングシステムを名前 (Chrome、Firefox) ごと、または名前とバージョン (Chrome 40.0、Firefox 35.0) ごとにグループ化するかどうかを指定します。
5. (ロケーショングラフのみ) [Granularity] で、グラフに 1 日につき 1 つのデータポイントを表示するか、1 時間につき 1 つのデータポイントを表示するかを指定します。14 日を超える日付範囲を指定した場合、1 時間につき 1 つのデータポイントを指定することはできなくなります。
6. (ロケーショングラフのみ) [Details] で、上位のロケーションを国ごとに表示するか、米国の州ごとに表示するかを指定します。
7. [Distribution] リストでは、使用状況グラフにデータを表示するディストリビューションを選択します。

   - 個々のディストリビューション - 選択した CloudFront ディストリビューションのデータがグラフに表示されます。[Distribution] リストには、ディストリビューションのディストリビューション ID と代替ドメイン名 (CNAME) が表示されます (ある場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

- All distributions (excludes deleted) - 現在の AWS アカウントに関連付けられているすべてのディストリビューションのデータが集計されてグラフに表示されます。ただし、削除したディストリビューションは除外されます。

8. [Update] (更新) を選択します。

グラフ内の毎日または毎時間のデータポイントのデータを表示するには、データポイントの上にカーソルを合わせます。

CSV 形式でデータをダウンロードする

各ビューワーレポートは CSV 形式でダウンロードできます。このセクションでは、レポートをダウンロードする方法と、レポートの値について説明します。

ビューワーレポートを CSV 形式でダウンロードするには

1. ビューワーレポートを表示しているときに、[CSV] を選択します。
2. ダウンロードするデータ ([Devices] や [Devices Trends] など) を選択します。
3. [Opening file name] ダイアログボックスで、ファイルを開くか保存するかを選択します。

ビューワーレポートに含まれるデータ

各レポートの先頭数行には次の情報が含まれます。

Version

この CSV ファイルの形式のバージョン。

レポート

レポートの名前。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

StartDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の開始日。

EndDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の終了日。

GeneratedTimeUTC

協定世界時 (UTC) によるレポートを実行した日時。

Grouping (ブラウザとオペレーティングシステムのレポートのみ)

データをブラウザやオペレーティングシステムの名前によってグループ化するか、または名前とバージョンによってグループ化するか。

詳細度

レポートの各行が 1 時間と 1 日のどちらを表すか。

Details (ロケーションレポートのみ)

リクエストを国別にリストするか、米国の州別にリストするか。

以下のトピックでは、各種ビューワーレポートの情報について説明します。

トピック

- デバイスレポート
- デバイストレンドレポート
- ブラウザレポート
- ブラウザトレンドレポート
- オペレーティングシステムレポート
- オペレーティングシステムトレンドレポート
- ロケーションレポート
- ロケーショントレンドレポート

デバイスレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

リクエスト

CloudFront が各タイプのデバイスから受け取ったリクエストの数。

RequestsPct

CloudFront がすべてのデバイスから受け取ったリクエストの数に対する、CloudFront が各タイプのデバイスから受け取ったリクエストの数の割合。

デバイストレンドレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は `ALL`。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

Desktop

期間中に CloudFront がデスクトップコンピュータから受け取ったリクエストの数。

モバイル

期間中に CloudFront がモバイルデバイスから受け取ったリクエストの数。モバイルデバイスには、タブレットと携帯電話の両方が含まれる場合があります。CloudFront でリクエストがモバイルデバイスとタブレットのいずれから発信されたかを特定できない場合、そのリクエストは `Mobile` 列でカウントされます。

Smart-TV

期間中に CloudFront がスマート TV から受け取ったリクエストの数。

Tablet

期間中に CloudFront がタブレットから受け取ったリクエストの数。CloudFront でリクエストがモバイルデバイスとタブレットのいずれから発信されたかを特定できない場合、そのリクエストは `Mobile` 列でカウントされます。

不明

`User-Agent` HTTP ヘッダーの値が標準デバイスタイプのいずれか (`Desktop` や `Mobile` など) に関連付けられていなかったリクエスト。

Empty

期間中に CloudFront が受け取ったリクエストで、HTTP `User-Agent` ヘッダーに値が含まれていなかったリクエストの数。

ブラウザレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は `ALL`。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

グループ

`Grouping` の値に応じて、CloudFront がリクエストを受け取ったブラウザまたはブラウザとバージョン。ブラウザ名に加えて、次の値が含まれる場合があります。

- Bot/Crawler – 主にコンテンツのインデックスを作成する検索エンジンからのリクエスト。
- Empty – `User-Agent` HTTP ヘッダーの値が空であったリクエスト。
- Other – CloudFront によって識別されたが、一般的ではないブラウザ。`Bot/Crawler`、`Empty`、`Unknown` が最初の 9 個の値に表示されない場合、`Other` にも含まれています。

- Unknown – User-Agent HTTP ヘッダーの値が標準的なブラウザに関連付けられていなかったリクエスト。このカテゴリのほとんどのリクエストは、カスタムアプリケーションまたはスクリプトからのリスクエストです。

リクエスト

CloudFront が各タイプのブラウザから受け取ったリクエストの数。

RequestsPct

期間中に CloudFront が受け取ったすべてのリクエストの数に対する、CloudFront が各タイプのブラウザから受け取ったリクエストの数の割合。

ブラウザトレンドレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

(ブラウザ)

レポートの残りの列には、Grouping の値に応じて、ブラウザまたはブラウザとバージョンがリストされます。ブラウザ名に加えて、次の値が含まれる場合があります。

- Bot/Crawler – 主にコンテンツのインデックスを作成する検索エンジンからのリクエスト。
- Empty – User-Agent HTTP ヘッダーの値が空であったリクエスト。
- Other – CloudFront によって識別されたが、一般的ではないブラウザ。Bot/Crawler、Empty、Unknown が最初の 9 個の値に表示されない場合、Other にも含まれています。

- Unknown – User-Agent HTTP ヘッダーの値が標準的なブラウザに関連付けられていなかったリクエスト。このカテゴリのほとんどのリクエストは、カスタムアプリケーションまたはスクリプトからのリスクエストです。

オペレーティングシステムレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

グループ

Grouping の値に応じて、CloudFront がリクエストを受け取ったオペレーティングシステムまたはオペレーティングシステムとバージョン。オペレーティングシステム名に加えて、次の値が含まれる場合があります。

- Bot/Crawler – 主にコンテンツのインデックスを作成する検索エンジンからのリクエスト。
- Empty – User-Agent HTTP ヘッダーの値が空であったリクエスト。
- Other – CloudFront によって識別されたが、一般的ではないオペレーティングシステム。Bot/Crawler、Empty、Unknown が最初の 9 個の値に表示されない場合、Other にも含まれています。
- Unknown – User-Agent HTTP ヘッダーの値が標準的なブラウザに関連付けられていなかったリクエスト。このカテゴリのほとんどのリクエストは、カスタムアプリケーションまたはスクリプトからのリスクエストです。

リクエスト

CloudFront が各タイプのオペレーティングシステムから受け取ったリクエストの数。

RequestsPct

期間中に CloudFront が受け取ったすべてのリクエストの数に対する、CloudFront が各タイプのオペレーティングシステムから受け取ったリクエストの数の割合。

オペレーティングシステムトレンドレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

(オペレーティングシステム)

レポートの残りの列には、Grouping の値に応じて、オペレーティングシステムまたはオペレーティングシステムとバージョンがリストされます。オペレーティングシステム名に加えて、次の値が含まれる場合があります。

- Bot/Crawler – 主にコンテンツのインデックスを作成する検索エンジンからのリクエスト。
- Empty – User-Agent HTTP ヘッダーの値が空であったリクエスト。
- Other – CloudFront によって識別されたが、一般的ではないオペレーティングシステム。Bot/Crawler、Empty、Unknown が最初の 9 個の値に表示されない場合、Other にも含まれています。
- Unknown – User-Agent HTTP ヘッダーでオペレーティングシステムが指定されていないリクエスト。

ロケーションレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

LocationCode

CloudFront が受け取ったリクエストの発信元のロケーションの略語。表示される可能性がある値の詳細については、「ロケーションレポートのデータと CloudFront 標準ログ (アクセスログ) のデータとの関連」のロケーションの説明を参照してください。

LocationName

CloudFront が受け取ったリクエストの発信元のロケーションの名前。

リクエスト

CloudFront が各ロケーションから受け取ったリクエストの数。

RequestsPct

期間中に CloudFront がすべてのロケーションから受け取ったリクエストの数に対する、CloudFront が各ロケーションから受け取ったリクエストの数の割合。

TotalBytes

指定したディストリビューションおよび期間に、CloudFront からこの国または州のビューワーに提供されたバイト数。

ロケーショントレンドレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

(ロケーション)

レポートの残りの列には、CloudFront が受け取ったリクエストの発信元のロケーションがリストされます。表示される可能性がある値の詳細については、「ロケーションレポートのデータと CloudFront 標準ログ (アクセスログ) のデータとの関連」のロケーションの説明を参照してください。

ロケーションレポートのデータと CloudFront 標準ログ (アクセスログ) のデータとの関連

次のリストは、CloudFront コンソールのロケーションレポートの値と、対応する CloudFront アクセスログの値を示します。CloudFront アクセスログの詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。

場所

ビューワーがいる国または米国の州。アクセスログの `c-ip` 列には、ビューワーが実行中のデバイスの IP アドレスが含まれています。位置情報データを使用して、IP アドレスに基づくデバイスの地理的場所を識別します。

[Locations] レポートを国ごとに表示している場合、国のリストは「ISO 3166-2、国および地方行政区画に対するコード – Part 2: 行政区画コード」を基にしている点に注意してください。国のリストには、以下の追加の値が含まれています。

- Anonymous Proxy (匿名プロキシ) – 匿名のプロキシからのリクエスト。
- Satellite Provider (衛星プロバイダー) – 複数の国にインターネットサービスを提供している衛星プロバイダーからのリクエスト。ビューワーは、不正行為のリスクが高い国にいる可能性があります。
- Europe (Unknown) (欧州 (不明)) – 複数の欧州諸国で使用されているブロックの IP からのリクエスト。リクエスト元の国を特定することはできません。CloudFront では、[Europe (Unknown) (欧州 (不明))] はデフォルトとして使用されます。
- Asia/Pacific (Unknown) (アジアパシフィック (不明)) – アジアパシフィックリージョンの複数の国で使用されているブロックの IP からのリクエスト。リクエスト元の国を特定することはできません。CloudFront では、[Asia/Pacific (Unknown) (アジアパシフィック (不明))] はデフォルトとして使用されます。

[Locations (所在地)] レポートを米国の州ごとに表示している場合、レポートには以下の米国準州と米軍基地所在地域が含まれることがあります。

> **Note**
>
> CloudFront がユーザーの所在地を特定できない場合、所在地はビューワーレポートに不明と表示されます。

Request Count

ビューワーがいる国または米国の州からの、指定したディストリビューションおよび期間のリクエストの総数。一般的にこの値は、CloudFront アクセスログのその国または州の IP アドレスからの GET リクエストの数とほぼ一致します。

リクエスト %

[Details] で選択した値に基づき、次のうちのいずれか。

- Countries – リクエストの総数に対するこの国からのリクエストの割合。
- U.S. States - 米国からのリクエストの総数に対するこの州からのリクエストの割合。

50 以上の国からリクエストがあると、[Request Count] 列に指定した期間のすべてのリクエストを含めることができないため、このテーブルのデータに基づいて [Request %] を計算することはできません。

バイト

指定したディストリビューションおよび期間に、CloudFront からこの国または州のビューワーに提供されたバイト数。この列のデータの表示を KB、MB、または GB に変更するには、列見出しのリンクをクリックします。

Amazon CloudWatch による CloudFront メトリクスのモニタリング

Amazon CloudFront は Amazon CloudWatch と統合されており、ディストリビューションの運用メトリクスとエッジ関数 (Lambda@Edge 関数と CloudFront Functions の両方) を自動的に公開します。これらのメトリクスの多くは、CloudFront コンソールの一連のグラフに表示され、CloudFront API または CLI を使用してアクセスすることもできます。これらのメトリクスはすべて、CloudWatch コンソールで、または CloudWatch API または CLI を介して利用できま

す。CloudFront メトリクスは CloudWatch のクォータ (以前は制限と呼ばれていました) にはカウントされません。また、追加料金も発生しません。

CloudFront ディストリビューションのデフォルトメトリクスに加えて、追加のメトリクスを追加料金で有効にすることができます。追加のメトリクスは CloudFront ディストリビューションに適用され、ディストリビューションごとに個別に有効にする必要があります。料金の詳細については、「the section called “追加の CloudFront メトリクスのコストの見積り”」を参照してください。

これらのメトリクスを表示すると、問題のトラブルシューティング、追跡、およびデバッグに役立ちます。CloudFront コンソールでこれらのメトリクスを表示するには、[Monitoring] (モニタリング) ページを参照してください。特定の CloudFront ディストリビューションまたはエッジ関数のアクティビティに関するグラフを表示するには、いずれかを選択してから、[View distribution metrics] (ディストリビューションのメトリクスの表示) または [View metrics] (メトリクスの表示) を選択します。

CloudFront コンソール、CloudWatch コンソール、API、または CLI で、これらのメトリクスに基づくアラームを設定することもできます (標準の CloudWatch 料金が適用されます)。例えば、`5xxErrorRate` メトリクスに基づくアラームを設定できます。このメトリクスは、レスポンスの HTTP ステータスコードが `500` から `599` の範囲内にあるすべてのビューワーリクエストの割合 (%) を示します。エラー率が一定時間内に特定の値 (連続した 5 分以内のリクエスト数の 5% など) に達すると、アラームがトリガーされます。アラームの作成時に、アラームの値と時間単位を指定します。詳細については、「アラームの作成」を参照してください。

> **Note**
>
> CloudFront コンソールで CloudWatch アラームを作成すると、アラームは米国東部 (バージニア北部) リージョン (`us-east-1`) に自動的に作成されます。CloudWatch コンソールでアラームを作成する場合は、同じリージョンを使用する必要があります。CloudFront はグローバルサービスであるため、サービスのメトリクスは米国東部 (バージニア北部) に送信されます。

トピック

- CloudFront 関数およびエッジ関数のメトリクスの表示
- メトリクスのアラームを作成する
- CSV 形式でのメトリクスデータのダウンロード
- CloudWatch API を使用したメトリクスの取得

CloudFront 関数およびエッジ関数のメトリクスの表示

CloudFront ディストリビューションとエッジ関数に関する運用メトリクスは CloudFront コンソールで表示できます。これらのメトリクスを表示するには、CloudFront コンソールの [Monitoring](モニタリング) ページを参照してください。特定の CloudFront ディストリビューションまたはエッジ関数のアクティビティに関するグラフを表示するには、いずれかを選択してから、[View distribution metrics] (ディストリビューションのメトリクスの表示) または [View metrics] (メトリクスの表示) を選択します。

トピック

- CloudFront ディストリビューションのデフォルトメトリクスの表示
- CloudFront ディストリビューションの追加のメトリクスの有効化
- Lambda@Edge 関数のデフォルトメトリクスの表示
- CloudFront Functions のデフォルトメトリクスの表示

CloudFront ディストリビューションのデフォルトメトリクスの表示

すべての CloudFront ディストリビューションについて、以下のデフォルトメトリクスが追加料金なしで表示されます。

リクエスト

すべての HTTP メソッド、および HTTP リクエストと HTTPS リクエストの両方について CloudFront が受信したビューワーリクエストの総数。

ダウンロードされたバイト数

GET リクエスト、HEAD リクエスト、および OPTIONS リクエストに対してビューワーがダウンロードしたバイト総数。

アップロードされたバイト数

ビューワーが POST リクエストと PUT リクエストを使用して CloudFront にアップロードしたバイトの総数。

4xx エラー率

レスポンスの HTTP ステータスコードが 4xx であるすべてのビューワーリクエストの割合 (%)。

5xx エラー率

レスポンスの HTTP ステータスコードが 5xx であるすべてのビューワーリクエストの割合 (%)。

合計エラー率

レスポンスの HTTP ステータスコードが 4xx または 5xx であるすべてのビューワーリクエストの割合 (%)。

これらのメトリクスは、CloudFront ディストリビューションごとにグラフとして、CloudFront コンソールの [Monitoring](モニタリング) ページに表示されます。各グラフでは、総数が 1 分単位で表示されます。グラフを表示するだけでなく、メトリクスレポートを CSV ファイルとしてダウンロードすることもできます。

次の手順を実行してグラフをカスタマイズできます。

- グラフに表示される情報の時間範囲を変更するには、1h (1 時間)、3h (3 時間)、または別の範囲、またはカスタムの範囲を指定します。
- CloudFront でグラフ内の情報を更新する頻度を変更するには、最新表示アイコンの横にある下矢印を選択してから、リフレッシュレートを選択します。デフォルトの更新間隔は 1 分ですが、10 秒、2 分、または他のオプションを指定しできます。

CloudWatch コンソールで CloudFront グラフを表示するには、[Add to dashboard] (ダッシュボードに追加) を選択します。

CloudFront ディストリビューションの追加のメトリクスの有効化

デフォルトメトリクスに加えて、追加のメトリクスを追加料金で有効にすることができます。料金の詳細については、「the section called “追加の CloudFront メトリクスのコストの見積り”」を参照してください。

以下の追加のメトリクスは、ディストリビューションごとに個別に有効にする必要があります。

キャッシュヒットレート

CloudFront がそのキャッシュからコンテンツを送信した対象のすべてのキャッシュ可能なリクエストの割合 (%)。HTTP POST/PUT リクエストおよびエラーは、キャッシュ可能なリクエストとは見なされません。

オリジンのレイテンシー

CloudFront キャッシュではなくオリジンから送信されたリクエストについて、CloudFront がリクエストを受信してからネットワーク (ビューワーではなく) にレスポンスを提供し始めるまでに費

やした合計時間。これは、最初のバイトのレイテンシーまたは最初のバイトまでの時間と呼ばれます。

ステータスコード別のエラー率

レスポンスの HTTP ステータスコードが 4xx 範囲または 5xx 範囲内の特定のコードであるすべてのビューワーリクエストの割合 (%)。このメトリクスは、401、403、404、502、503、および 504 のすべてのエラーコードで使用できます。

追加メトリクスの有効化

追加のメトリクスは、CloudFront コンソール、AWS CloudFormation、AWS Command Line Interface (AWS CLI)、または CloudFront API で有効にすることができます。

Console

追加のメトリクスを有効にするには (コンソール)

1. AWS Management Console にサインインして、CloudFront コンソールの [Monitoring](モニタリング) ページを開きます。
2. 追加のメトリクスを有効にするディストリビューションを選択し、[View distribution metrics] (ディストリビューションメトリクスの表示) を選択します。
3. [Manage additional metrics] (追加のメトリクスの管理) を選択します。
4. [Manage additional metrics] (追加のメトリクスの管理) ウィンドウで、[Enabled] (有効) をオンにします。追加のメトリクスを有効にしたら、[Manage additional metrics] (追加のメトリクスの管理) ウィンドウを閉じることができます。

有効にした追加のメトリクスがグラフに表示されます。各グラフでは、総数が 1 分単位で表示されます。グラフを表示するだけでなく、メトリクスレポートを CSV ファイルとしてダウンロードすることもできます。

次の手順を実行してグラフをカスタマイズできます。

- グラフに表示される情報の時間範囲を変更するには、1h (1 時間)、3h (3 時間)、または別の範囲、またはカスタムの範囲を指定します。
- CloudFront でグラフ内の情報を更新する頻度を変更するには、最新表示アイコンの横にある下矢印を選択してから、リフレッシュレートを選択します。デフォルトの更新間隔は 1 分ですが、10 秒、2 分、または他のオプションを指定しできます。

CloudWatch コンソールで CloudFront グラフを表示するには、[Add to dashboard] (ダッシュボードに追加) を選択します。

AWS CloudFormation

AWS CloudFormation で追加のメトリクスを有効にするには、`AWS::CloudFront::MonitoringSubscription` リソースタイプを使用します。次の例は、追加のメトリクスを有効にするための AWS CloudFormation テンプレート構文を YAML 形式で示しています。

```
Type: AWS::CloudFront::MonitoringSubscription
Properties:
  DistributionId: EDFDVBD6EXAMPLE
  MonitoringSubscription:
    RealtimeMetricsSubscriptionConfig:
      RealtimeMetricsSubscriptionStatus: Enabled
```

CLI

AWS Command Line Interface (AWS CLI) を使用して追加のメトリクスを管理するには、次のいずれかのコマンドを使用します。

ディストリビューション (CLI) の追加メトリクスを有効にするには

- 以下の例のように、create-monitoring-subscription コマンドを使用します。*EDFDVBD6EXAMPLE* を追加メトリクスを有効にするディストリビューションの ID と置き換えます。

```
aws cloudfront create-monitoring-subscription --
distribution-id EDFDVBD6EXAMPLE --monitoring-subscription
 RealtimeMetricsSubscriptionConfig={RealtimeMetricsSubscriptionStatus=Enabled}
```

ディストリビューション (CLI) の追加メトリクスが有効になっているかどうかを確認するには

- 以下の例のように、get-monitoring-subscription コマンドを使用します。*EDFDVBD6EXAMPLE* をチェックしているディストリビューションの ID と置き換えます。

```
aws cloudfront get-monitoring-subscription --distribution-id EDFDVBD6EXAMPLE
```

ディストリビューション (CLI) の追加メトリクスを無効にするには

- 以下の例のように、delete-monitoring-subscription コマンドを使用します。*EDFDVBD6EXAMPLE* を追加メトリクスを無効にするディストリビューションの ID と置き換えます。

```
aws cloudfront delete-monitoring-subscription --distribution-id EDFDVBD6EXAMPLE
```

API

CloudFront API を使用して追加メトリクスを管理するには、次の API オペレーションのいずれかを使用します。

- ディストリビューションの追加メトリクスを有効にするには、CreateMonitoringSubscription を使用します。
- ディストリビューションの追加メトリクスが有効になっているかどうかを確認するには、GetMonitoringSubscription を使用します。
- ディストリビューションの追加メトリクスを無効にするには、DeleteMonitoringSubscription を使用します。

これらの API コールの詳細については、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

追加の CloudFront メトリクスのコストの見積り

ディストリビューションの追加のメトリクスを有効にすると、CloudFront は最大 8 つのメトリクスを米国東部 (バージニア北部) リージョンで CloudWatch に送信します。CloudWatch では、メトリクスごとに低い固定料金が請求されます。この料金は、メトリクスごとに毎月 1 回のみ請求されます (ディストリビューションごとに最大 8 つのメトリクス)。これは固定レートであるため、CloudFront ディストリビューションが受信または送信するリクエストまたはレスポンスの数に関係なく、コストは同じままです。メトリクスごとの料金については、Amazon CloudWatch 料金ページと CloudWatch 料金計算ツールを参照してください。CloudWatch API を使用してメトリクスを取得すると、追加の API 料金が適用されます。

Lambda@Edge 関数のデフォルトメトリクスの表示

CloudWatch メトリクスを使用して、Lambda@Edge 関数の問題をリアルタイムでモニタリングできます。これらのメトリクスに対する追加料金はありません。

Lambda@Edge 関数を CloudFront ディストリビューションのキャッシュ動作にアタッチすると、Lambda はメトリクスを CloudWatch に自動的に送信し始めます メトリクスはすべての Lambda リージョンで利用可能ですが、CloudWatch コンソールでメトリクスを表示したり、CloudWatch API からメトリクスデータを取得したりするには、米国東部 (バージニア北部) リージョン (`us-east-1`) を使用する必要があります。メトリクスグループ名は `AWS/CloudFront/`*`distribution-ID`* の形式になります。ここで、*`distribution-ID`* は Lambda@Edge 関数が関連付けられている CloudFront ディストリビューションの ID です。Amazon CloudWatch メトリクスの詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

デフォルトメトリクスは、Lambda@Edge 関数ごとにグラフとして、CloudFront コンソールの [Monitoring](モニタリング) ページに表示されます。

- `5xx`Lambda@Edge の エラー率
- Lambda 実行エラー
- Lambda 無効レスポンス
- Lambda スロットリング

グラフには、呼び出し数、エラー数、スロットル数などが表示されます。各グラフでは、合計が 1 分単位で AWS リージョンごとにグループ化されて表示されます。

調査したいエラーが急増した場合は、問題が発生している関数と AWS リージョンを特定するまで、各関数を選択して AWS リージョン別にログファイルを表示できます。Lambda@Edge エラーのトラブルシューティングの詳細については、以下を参照してください。

- the section called “障害のタイプを判断する方法”
- AWS でコンテンツ配信をデバッグする 4 つのステップ

次の手順を実行してグラフをカスタマイズできます。

- グラフに表示される情報の時間範囲を変更するには、1h (1 時間)、3h (3 時間)、または別の範囲、またはカスタムの範囲を指定します。

- CloudFront でグラフ内の情報を更新する頻度を変更するには、最新表示アイコンの横にある下矢印を選択してから、リフレッシュレートを選択します。デフォルトの更新間隔は 1 分ですが、10 秒、2 分、または他のオプションを指定しできます。

CloudWatch コンソールにグラフを表示するには、[ダッシュボードに追加] を選択します。CloudWatch コンソールでグラフを表示するには、米国東部 (バージニア北部) リージョン (us-east-1) を使用する必要があります。

CloudFront Functions のデフォルトメトリクスの表示

CloudFront Functions は Amazon CloudWatch に運用メトリクスを送信し、関数をモニタリングできるようにします。これらのメトリクスを表示すると、問題のトラブルシューティング、追跡、およびデバッグに役立ちます。CloudFront Functions は、CloudWatch に次のメトリクスを公開します。

- 呼び出し (FunctionInvocations) - 指定された期間に関数が起動 (呼び出し) された回数。
- 検証エラー (FunctionValidationErrors) - 指定した期間内に関数によって生成された検証エラーの数。検証エラーは、関数は正常に実行されたが、無効なデータ (無効なイベントオブジェクト) を返した場合に発生します。
- 実行エラー (FunctionExecutionErrors) - 特定の期間に発生した実行エラーの数。実行エラーは、関数が正常に完了しなかった場合に発生します。
- コンピューティング使用率 (FunctionComputeUtilization) - 関数の実行にかかった時間 (最大許容時間に対するパーセンテージ)。たとえば、値 35 は、関数が最大許容時間の 35% で完了したことを意味します。このメトリクスは、0から100までの数値です。

  この値が 100 に達するか、100 に近い場合、関数は許容実行時間を使い切ったか、それに近い状態であるため、以降のリクエストはスロットリングされる可能性があります。関数の使用率が 80% 以上になっている場合は、関数を見直して実行時間を短縮し、使用率を向上させることをお勧めします。例えば、エラーのみを記録したり、複雑な正規表現を単純化したり、複雑な JSON オブジェクトの不要な解析を除外したりできます。
- スロットリング (FunctionThrottles) - 指定された期間に関数がスロットリングされた回数。関数は、次の理由でスロットリングできます。
  - この関数は、実行に許容される最大時間を継続的に超えている
  - この関数によってコンパイルエラーが発生する
  - 1 秒あたりのリクエスト数が異常に多い

CloudFront KeyValueStore は、以下のオペレーションメトリクスも Amazon CloudWatch に送信します。

- 読み取りリクエスト (KvsReadRequests) - 一定期間に関数がキー値ストアから正常に読み取った回数。
- 読み取りエラー (KvsReadErrors) - 一定期間に関数がキー値ストアからの読み取りに失敗した回数。

CloudFront コンソールでこれらのメトリクスを表示するには、[Monitoring page] を参照してください。特定の関数のグラフを表示するには、[Functions] で関数を選択した後、[View function metrics] を選択します。

これらのメトリクスはすべて、米国東部 (バージニア北部) リージョン (us-east-1) の CloudWatch に CloudFront 名前空間で発行されます。CloudWatch コンソールでこれらのメトリクスを表示することもできます。CloudWatch コンソールでは、関数ごと、またはディストリビューションごとの関数ごとのメトリクスを表示できます。

CloudWatch を使用して、これらのメトリクスに基づいたアラームを設定することもできます。例えば、実行時間 (FunctionComputeUilization) メトリクスに基づいてアラームを設定できます。このメトリクスは、関数の実行にかかった許容時間の割合を表します。実行時間が一定時間内に特定の値 (許容時間の 70% 以上を 15 分継続して超過など) に達すると、アラームがトリガーされます。アラームの作成時に、アラームの値と時間単位を指定します。

> **Note**
>
> CloudFront Functions は、本番リクエストとレスポンスに応答して実行されるステージ LIVE の関数のみ、メトリクスを CloudWatch に送信します。関数をテストするとき、CloudFront は CloudWatch にメトリクスを送信しません。テスト出力には、エラー、コンピューティング使用率、関数ログ (console.log() ステートメント) に関する情報が含まれていますが、この情報は CloudWatch に送信されません。

CloudWatch API でメトリクスを取得する方法についての詳細は、「the section called “API を使用したメトリクスの取得”」を参照してください。

メトリクスのアラームを作成する

CloudFront コンソールで、CloudFront の特定のメトリクスに基づいて Amazon Simple Notification Service (Amazon SNS) から通知を受け取るようにアラームを設定できます。アラームは、CloudFront コンソールの [Alarms](アラーム) ページで設定できます。

コンソールでアラームを作成するには、以下の値を指定します。

メトリクス

アラームを作成する対象のメトリクス。

配信

アラームを作成する対象の CloudFront ディストリビューション。

アラーム名

アラームの名前。

通知の送信先

このメトリクスがアラームをトリガーした場合に通知を送信する先の Amazon SNS トピック。

Whenever ***<metric> <operator> <value>***

どのような場合に CloudWatch でアラームをトリガーして Amazon SNS トピックに通知を送信するかを指定します。たとえば、5xx エラー率が 1% を超えた場合に通知を受け取るには、次のように指定します。

Whenever Average of 5xxErrorRate **>** **1**

値を指定する際には、以下の点に注意します。

- 区切り文字を使用せず、整数のみを入力します。たとえば、1,000 を指定する場合は、「**1000**」と入力します。
- 4xx、5xx、および合計エラー率の場合、指定する値は割合 (%) です。
- リクエスト、ダウンロードされたバイト数、アップロードされたバイト数の場合、指定する値は単位です。たとえば、1073742000 バイトと指定します。

For at least ***<number>*** consecutive periods of ***<time period>***

メトリクスが条件を満たした指定の期間が連続して何回続いたときに、CloudWatch でアラームをトリガーするかを指定します。値を選択するときは、一時的または瞬間的な問題に対してはア

ラームをトリガーしない値と、持続的な問題や実際の問題に対してはアラームをトリガーする値の間で適切なバランスを取ります。

CSV 形式でのメトリクスデータのダウンロード

CloudFront ディストリビューションの CloudWatch メトリクスデータを CSV 形式でダウンロードできます。CloudFront コンソールで特定のディストリビューションのディストリビューションメトリクスを表示する場合は、データをダウンロードできます。

レポートに関する情報

レポートの先頭数行には次の情報が含まれます。

バージョン

CloudFront レポート機能のバージョン。

レポート

レポートの名前。

DistributionID

レポートを実行した対象のディストリビューションの ID。

StartDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の開始日。

EndDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の終了日。

GeneratedTimeUTC

協定世界時 (UTC) によるレポートを実行した日時。

詳細度

レポートの行ごとの期間 (`ONE_MINUTE` など)。

メトリクスレポートのデータ

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

リクエスト

該当期間中のすべての HTTP ステータスコード (200、404 など) およびすべてのメソッド (GET、HEAD、POST など) に対するリクエストの総数。

BytesDownloaded

期間中に指定したディストリビューションについてビューワーがダウンロードしたバイト数。

BytesUploaded

指定したディストリビューションについて、該当期間中にビューワーがダウンロードしたバイト数。

TotalErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 4xx または 5xx エラーであったリクエストの割合 (%)。

4xxErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 4xx エラーであったリクエストの割合 (%)。

5xxErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 5xx エラーであったリクエストの割合 (%)。

ディストリビューションの追加のメトリクスを有効化している場合、レポートには以下の追加の値も表示されます。

401ErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 401 エラーであったリクエストの割合 (%)。

403ErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 403 エラーであったリクエストの割合 (%)。

404ErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 404 エラーであったリクエストの割合 (%)。

502ErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 502 エラーであったリクエストの割合 (%)。

503ErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 503 エラーであったリクエストの割合 (%)。

504ErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 504 エラーであったリクエストの割合 (%)。

OriginLatency

CloudFront キャッシュではなくオリジンから送信されたリクエストについて、CloudFront がリクエストを受信してからネットワーク (ビューワーではなく) にレスポンスを提供し始めるまでに費やした合計時間 (ミリ秒単位)。これは、最初のバイトのレイテンシーまたは最初のバイトまでの時間と呼ばれます。

CacheHitRate

CloudFront がそのキャッシュからコンテンツを送信した対象のすべてのキャッシュ可能なリクエストの割合 (%)。HTTP POST/PUT リクエストおよびエラーは、キャッシュ可能なリクエストとは見なされません。

CloudWatch API を使用したメトリクスの取得

Amazon CloudWatch API または CLI を使用して、構築したプログラムやアプリケーションで CloudFront メトリクスを取得できます。raw データを使用して、独自のカスタムダッシュボードや独自のアラームツールなどを構築できます。

CloudWatch API から CloudFront メトリクスを取得するには、米国東部 (バージニア北部) リージョン (`us-east-1`) を使用する必要があります。また、各メトリクスの特定の値とタイプも知っておく必要があります。

トピック

すべての CloudFront メトリクスの値

以下の値は、すべての CloudFront メトリクスに適用されます。

`Namespace`

`Namespace` の値は常に `AWS/CloudFront` です。

ディメンション

各 CloudFront メトリクスには、次の 2 つのディメンションがあります。

`DistributionId`

メトリクスを取得する対象の CloudFront ディストリビューションの ID。

`FunctionName`

メトリクスを取得する対象の (CloudFront Functions の) 関数の名前。

このディメンジョンは、関数にのみ適用されます。

`Region`

`Region` の値は常に `Global` です。CloudFront はグローバルサービスであるためです。

> **Note**
>
> CloudWatch API から CloudFront メトリクスを取得するには、米国東部 (バージニア北部) リージョン (us-east-1) を使用する必要があります。

CloudFront ディストリビューションメトリクスの値

次のリストの情報を使用して、CloudWatch API から特定の CloudFront ディストリビューションメトリクスに関する詳細を取得します。これらのメトリクスの一部は、ディストリビューションで追加のメトリクスを有効にしている場合にのみ使用できます。

> **Note**
>
> 各メトリクスには 1 つの統計 (Average または Sum) のみを適用できます。次のリストは、各メトリクスに適用できる統計を示しています。

4xx エラー率

レスポンスの HTTP ステータスコードが 4xx であるすべてのビューワーリクエストの割合 (%)。

- メトリクス名: 4xxErrorRate
- 有効な統計: Average
- 単位: Percent

401 エラー率

レスポンスの HTTP ステータスコードが 401 であるすべてのビューワーリクエストの割合 (%)。このメトリクスを取得するには、まず追加のメトリクスを有効にする必要があります。

- メトリクス名: 401ErrorRate
- 有効な統計: Average
- 単位: Percent

403 エラー率

レスポンスの HTTP ステータスコードが 403 であるすべてのビューワーリクエストの割合 (%)。このメトリクスを取得するには、まず追加のメトリクスを有効にする必要があります。

- メトリクス名: 403ErrorRate

- 有効な統計: Average
- 単位: Percent

404 エラー率

レスポンスの HTTP ステータスコードが 404 であるすべてのビューワーリクエストの割合 (%)。このメトリクスを取得するには、まず追加のメトリクスを有効にする必要があります。

- メトリクス名: 404ErrorRate
- 有効な統計: Average
- 単位: Percent

5xx エラー率

レスポンスの HTTP ステータスコードが 5xx であるすべてのビューワーリクエストの割合 (%)。

- メトリクス名: 5xxErrorRate
- 有効な統計: Average
- 単位: Percent

502 エラー率

レスポンスの HTTP ステータスコードが 502 であるすべてのビューワーリクエストの割合 (%)。このメトリクスを取得するには、まず追加のメトリクスを有効にする必要があります。

- メトリクス名: 502ErrorRate
- 有効な統計: Average
- 単位: Percent

503 エラー率

レスポンスの HTTP ステータスコードが 503 であるすべてのビューワーリクエストの割合 (%)。このメトリクスを取得するには、まず追加のメトリクスを有効にする必要があります。

- メトリクス名: 503ErrorRate
- 有効な統計: Average
- 単位: Percent

504 エラー率

レスポンスの HTTP ステータスコードが 504 であるすべてのビューワーリクエストの割合 (%)。このメトリクスを取得するには、まず追加のメトリクスを有効にする必要があります。

- メトリクス名: 504ErrorRate
- 有効な統計: Average
- 単位: Percent

ダウンロードされたバイト数

GET リクエスト、HEAD リクエスト、および OPTIONS リクエストに対してビューワーがダウンロードしたバイト総数。

- メトリクス名: BytesDownloaded
- 有効な統計: Sum
- 単位: None

アップロードされたバイト数

POST リクエストと PUT リクエストを使用して CloudFront でビューワーがオリジンにアップロードしたバイト総数。

- メトリクス名: BytesUploaded
- 有効な統計: Sum
- 単位: None

キャッシュヒットレート

CloudFront がそのキャッシュからコンテンツを送信した対象のすべてのキャッシュ可能なリクエストの割合 (%)。HTTP POST/PUT リクエストおよびエラーは、キャッシュ可能なリクエストとは見なされません。このメトリクスを取得するには、まず追加のメトリクスを有効にする必要があります。

- メトリクス名: CacheHitRate
- 有効な統計: Average
- 単位: Percent

オリジンのレイテンシー

CloudFront キャッシュではなくオリジンから送信されたリクエストについて、CloudFront がリクエストを受信してからネットワーク (ビューワーではなく) にレスポンスを提供し始めるまでに費やした合計時間 (ミリ秒単位)。これは、最初のバイトのレイテンシーまたは最初のバイトまでの時間と呼ばれます。このメトリクスを取得するには、まず追加のメトリクスを有効にする必要があります。

- メトリクス名: OriginLatency

- 有効な統計: Percentile
- 単位: Milliseconds

> **Note**
>
> CloudWatch API から Percentile 統計情報を取得するには、ExtendedStatistics ではなく Statistics パラメータを使用します。詳細については、Amazon CloudWatch API リファレンスの「GetMetricStatistics」、または AWS SDK のリファレンスドキュメントを参照してください。

リクエスト

すべての HTTP メソッド、および HTTP リクエストと HTTPS リクエストの両方について CloudFront が受信したビューワーリクエストの総数。

- メトリクス名: Requests
- 有効な統計: Sum
- 単位: None

合計エラー率

レスポンスの HTTP ステータスコードが 4xx または 5xx であるすべてのビューワーリクエストの割合 (%)。

- メトリクス名: TotalErrorRate
- 有効な統計: Average
- 単位: Percent

CloudFront 関数メトリクスの値

次のリストの情報を使用して、CloudWatch API から特定の CloudFront 関数メトリクスに関する詳細を取得します。

> **Note**
>
> 各メトリクスには 1 つの統計 (Average または Sum) のみを適用できます。次のリストは、各メトリクスに適用できる統計を示しています。

呼び出し

指定された期間に関数が起動 (呼び出し) された回数。

- メトリクス名: FunctionInvocations
- 有効な統計: Sum
- 単位: None

検証エラー

指定された期間に関数によって生成された検証エラーの数。検証エラーは、関数は正常に実行されたが、無効なデータ (無効なイベントオブジェクト) を返した場合に発生します。

- メトリクス名: FunctionValidationErrors
- 有効な統計: Sum
- 単位: None

実行エラー

特定の期間に発生した実行エラーの数。実行エラーは、関数が正常に完了しなかった場合に発生します。

- メトリクス名: FunctionExecutionErrors
- 有効な統計: Sum
- 単位: None

コンピューティング使用率

関数の実行にかかった時間の長さ (0-100) を、最大許容時間に対するパーセンテージで示します。たとえば、値 35 は、関数が最大許容時間の 35% で完了したことを意味します。

- メトリクス名: FunctionComputeUtilization
- 有効な統計: Average
- 単位: Percent

Throttles

指定された期間に関数がスロットリングされた回数。

- メトリクス名: FunctionThrottles
- 有効な統計: Sum

- 単位: None

CloudFront とエッジ関数のログ記録

Amazon CloudFront では、さまざまな種類のログ記録が提供されます。CloudFront ディストリビューションに送信されるビューワーリクエスト、または AWS アカウントの CloudFront サービスアクティビティ (API アクティビティ) をログに記録することができます。エッジコンピューティング関数からログを取得することもできます。

リクエストのログ記録

CloudFront には、ディストリビューションに送信されるリクエストをログに記録するために、次の方法が用意されています。

標準ログ (アクセスログ)

CloudFront 標準ログは、ディストリビューションに対して行われたすべてのリクエストに関する詳細なレコードを提供します。これらのログは、セキュリティやアクセスの監査などの多くのシナリオに役立ちます。

CloudFront 標準ログは、選択した Amazon S3 バケットに配信されます。CloudFront で標準ログの料金は発生しませんが、ログファイルの保存とアクセスについては Amazon S3 の料金が発生します。

詳しくは、「標準ログ (アクセスログ) の使用」を参照してください。

リアルタイムログ

CloudFront リアルタイムログは、ディストリビューションに対して行われたリクエストに関する情報をリアルタイムで提供します (ログレコードはリクエストを受信してから数秒以内に配信されます)。リアルタイムログのサンプリングレート、つまり、リアルタイムのログ記録を受信するリクエストの割合を選択できます。ログ記録で受信が行われる特定のフィールドを選択することもできます。

CloudFront リアルタイムログは、Amazon Kinesis Data Streams で選択したデータストリームに配信されます。Kinesis Data Streams の使用料金に加えて、CloudFront でのリアルタイムログの料金が発生します。

詳しくは、「リアルタイムログ」を参照してください。

エッジ関数を口グ記録する

Amazon CloudWatch Logs を使用して、Lambda@Edge 関数と CloudFront Functions の両方のエッジ関数のログを取得できます。ログには、CloudWatch コンソールまたは CloudWatch Logs API を使用してアクセスできます。詳しくは、「the section called “エッジ関数のログ”」を参照してください。

サービスアクティビティのログ記録

AWS CloudTrail を使用して、AWS アカウントの CloudFront サービスアクティビティ (API アクティビティ) をログに記録できます。CloudTrail は、CloudFront のユーザー、ロール、または AWS のサービスによって実行された API アクションの記録を提供します。CloudTrail で収集された情報を使用して、CloudFront に対する API リクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

詳細については、「AWS CloudTrail を使用した Amazon CloudFront API コールのログ記録」を参照してください。

トピック

- 標準ログ (アクセスログ) の設定および使用
- リアルタイムログ
- エッジ関数のログ
- AWS CloudTrail を使用した Amazon CloudFront API コールのログ記録

標準ログ (アクセスログ) の設定および使用

CloudFront が受信するすべてのユーザーリクエストに関する詳細情報を含めたログファイルが作成されるように CloudFront を設定できます。これらは、標準ログと呼ばれます。また、アクセスログとも呼ばれています。標準ログを有効にする場合、CloudFront でファイルを保存する Amazon S3 バケットも指定できます。

ディストリビューションを作成または更新するとき、標準ログを有効にできます。詳細については、「ディストリビューション設定リファレンス」を参照してください。

CloudFront は、リアルタイムログも提供します。これにより、ディストリビューションに対して行われたリクエストに関する情報がリアルタイムで提供されます (ログはリクエストを受信してから数秒以内に配信されます)。リアルタイムログを使用して、コンテンツ配信のパフォーマンスに基づい

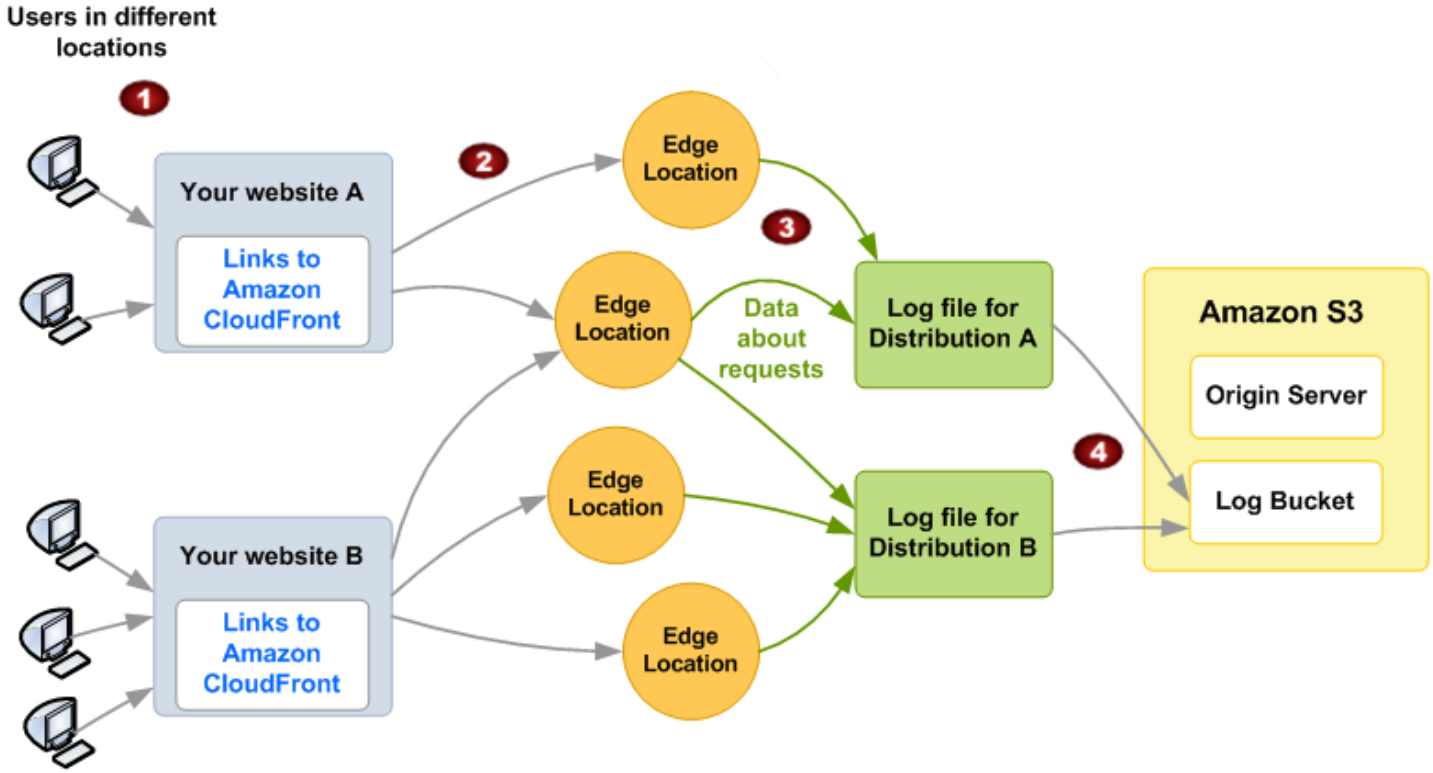
て監視、分析、アクションを実行できます。詳細については、「リアルタイムログ」を参照してください。

トピック

- 標準ログ記録のしくみ
- 標準ログ用の Amazon S3 バケットの選択
- 標準ログ記録の設定およびログファイルへのアクセスに必要なアクセス許可
- SSE-KMS バケット必須のキーポリシー
- ファイル名の形式
- 標準ログファイル配信のタイミング
- リクエスト URL またはヘッダーが最大のサイズを超えた場合にリクエストがどのようにログに記録されるか
- 標準ログの分析
- 標準ログ記録設定の編集
- Amazon S3 バケットからの標準ログファイルの削除
- 標準ログファイル形式
- 標準ログの料金

標準ログ記録のしくみ

次の図は、オブジェクトのリクエストに関する情報が CloudFront によってログ記録されるしくみを示しています。



以下では、前の図に示すように、オブジェクトのリクエストに関する情報が CloudFront によって口グ記録されるしくみを説明します。

1. この図には、2 つのウェブサイト (A、B) と、対応する 2 つの CloudFront ディストリビューションが示されています。ユーザーは、ディストリビューションに関連付けられている URL を使用してオブジェクトをリクエストします。
2. CloudFront は、各リクエストを適切なエッジロケーションにルーティングします。
3. CloudFront は、各リクエストに関するデータを、そのディストリビューション専用のログファイルに書き込みます。この例では、ディストリビューション A に関連するリクエストについての情報がディストリビューション A 専用のログファイルに、ディストリビューション B に関連するリクエストについての情報がディストリビューション B 専用のログファイルに書き込まれます。
4. ログ記録を有効にした際に指定した Amazon S3 バケットに、ディストリビューションのログファイルが CloudFront によって定期的に保存されます。後続のリクエストに関する情報は、CloudFront によってディストリビューションの新しいログファイルに保存されます。

一定の時間、お客様のコンテンツに対してユーザーアクセスがない場合、その時間のログファイルを受け取ることはありません。

ログファイルには、1 つのリクエストの詳細が 1 エントリとして記録されます。ログファイル形式の詳細については、「標準ログファイル形式」を参照してください。

> **Note**
>
> ログは、すべてのリクエストを完全に課金するためのものではなく、コンテンツに対するリクエストの本質を把握するものとして使用することをお勧めします。CloudFront はベストエフォートベースでアクセスログを提供します。特定のリクエストのログエントリが、リクエストが実際に処理されてからかなり後に配信されることも、(まれに) 一切配信されないこともあります。ログエントリをアクセスログから省略すると、アクセスログ内のエントリ数は AWS の請求と使用状況レポートに表示される使用量と一致しなくなります。

標準ログ用の Amazon S3 バケットの選択

ディストリビューションのログ記録を有効にする際には、CloudFront でログファイルを保存する Amazon S3 バケットを指定します。オリジンとして Amazon S3 を使用する場合は、同じバケットをログファイルに使用しないことをお勧めします。別々のバケットを使用すると、メンテナンスが容易になります。

> **Important**
>
> 強制を行ったバケット所有者に設定される S3 オブジェクトの所有権を使用して Amazon S3 バケットを選択してはいけません。この設定では、バケットとその中のオブジェクトの ACL が無効になり、CloudFront によるバケットへのログファイルの配信を阻止します。

> **Important**
>
> 以下のいずれのリージョンでも Amazon S3 バケットを選択しないでください。CloudFront は、これらのリージョンのバケットに標準ログを配信しません。
>
> - アフリカ (ケープタウン)
> - アジアパシフィック (香港)
> - アジアパシフィック (ハイデラバード)
> - アジアパシフィック (ジャカルタ)
> - アジアパシフィック (メルボルン)

- カナダ西部 (カルガリー)
- 欧州 (ミラノ)
- 欧州 (スペイン)
- 欧州 (チューリッヒ)
- イスラエル (テルアビブ)
- 中東 (バーレーン)
- 中東 (アラブ首長国連邦)

複数のディストリビューションのログファイルを同じバケットに保存することもできます。ログ記録を有効にする際には、ファイル名のプレフィックスをオプションで指定できます。これにより、どのログファイルがどのディストリビューションに関連しているか追跡できます。

標準ログ記録の設定およびログファイルへのアクセスに必要なアクセス許可

Important

2023 年 4 月以降、CloudFront 標準ログに使用される新しい S3 バケットの S3 アクセスコントロールリスト (ACL) を有効にする必要があります。ACL は、バケット作成のステップ中、またはバケットが作成された後に有効にできます。
変更の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「新しい S3 バケットのデフォルト設定に関するよくある質問」と、「AWS ニュースブログ」の「注意喚起: 2023 年 4 月に予定されている Amazon S3 のセキュリティ変更」を参照してください。

AWS アカウントには、ログファイル用に指定するバケットに対する以下の許可が必要です。

- バケットの S3 アクセスコントロールリスト (ACL) は FULL_CONTROL を付与する必要があります。バケット所有者のアカウントには、デフォルトでこのアクセス許可があります。権限がない場合、バケット所有者はバケットの ACL を更新する必要があります。
- `s3:GetBucketAcl`
- `s3:PutBucketAcl`

次の点に注意してください。

バケットの ACL

ディストリビューションを作成または更新してロギングを有効にすると、CloudFront はこれらのアクセス許可を使用してバケットの ACL を更新し、awslogsdelivery アカウントに FULL_CONTROL のアクセス許可を付与します。awslogsdelivery アカウントはログファイルをバケットに書き込みます。アカウントに ACL を更新するために必要なアクセス許可がない場合、ディストリビューションの作成または更新は失敗します。

状況によっては、バケットを作成するリクエストをプログラムで送信したが、指定した名前のバケットが既に存在する場合、S3 ではバケットのアクセス許可をデフォルト値にリセットします。アクセスログを S3 バケットに保存するように CloudFront を設定した後で、そのバケットでのログ受信を中止する場合は、バケットのアクセス許可をチェックして CloudFront に必要なアクセス許可があることを確認します。

バケットの ACL を復元する

awslogsdelivery アカウントのアクセス許可を削除すると、CloudFront はログを S3 バケットに保存できません。CloudFront がディストリビューションのログ保存を再開するには、次のいずれかの操作を行って ACL アクセス許可を復元します。

- CloudFront ディストリビューションのログ記録を無効化してから、再度有効にします。詳細については、「ディストリビューション設定リファレンス」を参照してください。
- Amazon S3 コンソールで S3 バケットに移動してアクセス許可を追加することで、awslogsdelivery の ACL アクセス許可を手動で追加します。awslogsdelivery の ACL を追加するには、アカウントの正規 ID を入力する必要があります。これは次のとおりです。

  ```
  c4c1ede66af53448b93c283ce9448c4ba468c9432aa01d700d3878632f77d2d0
  ```

  ACL を S3 バケットに追加する方法の詳細については、Amazon Simple Storage Service ユーザーガイドの「ACL バケットの許可を設定する方法」を参照してください。

各ログファイルの ACL

バケットの ACL に加えて、各ログファイルの ACL があります。バケット所有者にはログファイルに対する FULL_CONTROL アクセス許可があり、ディストリビューション所有者 (バケット所有者と異なる場合) にはアクセス許可がありません。awslogsdelivery アカウントには読み取りアクセス許可と書き込みアクセス許可があります。

ログ記録の無効化

ログ記録を無効にしても、CloudFront ではバケットやログファイルの ACL が削除されません。これはお客様自身で行うことができます。

SSE-KMS バケット必須のキーポリシー

標準ログ用の S3 バケットで、カスタマーマネージド型キーを使用する AWS KMS keys (SSE-KMS) を用いたサーバー側の暗号化が使用されている場合は、カスタマーマネージド型キーのキーポリシーに次のステートメントを追加する必要があります。これにより、CloudFront はログファイルをバケットに書き込むことができます。(CloudFront はログファイルをバケットに書き込むことができないため、AWS マネージドキーで SSE-KMS を使用することはできません)

```
{
    "Sid": "Allow CloudFront to use the key to deliver logs",
    "Effect": "Allow",
    "Principal": {
        "Service": "delivery.logs.amazonaws.com"
    },
    "Action": "kms:GenerateDataKey*",
    "Resource": "*"
}
```

標準ログの S3 バケットで S3 バケットキー を有する SSE-KMS を使用する場合は、ポリシーステートメントに `kms:Decrypt` 許可を追加する必要もあります。この場合、完全なポリシーステートメントは次のようになります。

```
{
    "Sid": "Allow CloudFront to use the key to deliver logs",
    "Effect": "Allow",
    "Principal": {
        "Service": "delivery.logs.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey*",
        "kms:Decrypt"
    ],
    "Resource": "*"
}
```

ファイル名の形式

CloudFront が Amazon S3 バケットに保存する各ログファイルの名前には、次のファイル名形式が使用されます。

<optional prefix>/*<distribution ID>*.*YYYY*-*MM*-*DD*-*HH*.*unique-ID*.gz

日付と時刻は協定世界時 (UTC) です。

たとえば、example-prefix をプレフィックスとして使用している場合に、ディストリビューション ID が EMLARXS9EXAMPLE であれば、ファイル名は次のようになります。

example-prefix/EMLARXS9EXAMPLE.2019-11-14-20.RT4KCN4SGK9.gz

ディストリビューションのログ記録を有効にする際には、ファイル名のプレフィックスをオプションで指定できます。これにより、どのログファイルがどのディストリビューションに関連しているか追跡できます。ログファイルのプレフィックスの値を指定した場合、プレフィックスがスラッシュ (/) で終わらない場合は、CloudFront によって自動的に追加されます。プレフィックスがスラッシュで終わる場合、CloudFront はスラッシュを追加しません。

ファイル名の末尾にある .gz は、CloudFront によってログファイルが gzip で圧縮されたことを示しています。

標準ログファイル配信のタイミング

CloudFront は、ディストリビューションの標準ログを 1 時間に最大で数回配信します。一般的に、ログファイルには、一定期間内に CloudFront が受信したリクエストに関する情報が含まれています。CloudFront は通常、その期間のログファイルを、ログに書き込まれたイベントの発生から 1 時間以内に Amazon S3 バケットに配信します。ただし、ある期間のログファイルエントリの一部またはすべてが、最大で 24 時間遅れることもあります。ログエントリが遅れた場合、CloudFront はこれらをログファイルに保存します。そのファイル名には、ファイルが配信された日時ではなく、リクエストが発生した期間の日時が含められます。

CloudFront は、ログファイルを作成する場合、ログファイルに対応する期間中にオブジェクトについてリクエストを受信したすべてのエッジロケーションから、ディストリビューションの情報を集約します。

CloudFront は、ディストリビューションに関連付けられているオブジェクトについて CloudFront が受信したリクエストの数により、1 つの期間に対して複数のファイルを保存することもできます。

CloudFront によるアクセスログの出力が確実に行われるのは、ログ記録が有効になって約 4 時間後からです。この時間以前にも少しのアクセスログを取得できる場合もあります。

> **Note**
>
> 期間中にオブジェクトに対してユーザーによるリクエストがなければ、その期間のログファイルは配信されません。

CloudFront は、リアルタイムログも提供します。これにより、ディストリビューションに対して行われたリクエストに関する情報がリアルタイムで提供されます (ログはリクエストを受信してから数秒以内に配信されます)。リアルタイムログを使用して、コンテンツ配信のパフォーマンスに基づいて監視、分析、アクションを実行できます。詳細については、「リアルタイムログ」を参照してください。

リクエスト URL またはヘッダーが最大のサイズを超えた場合にリクエストがどのようにログに記録されるか

クッキーを含むすべてのリクエストヘッダーの合計サイズが 20 KB を超える場合、または URL が 8192 バイトの URL サイズ制限を超える場合、CloudFront ではリクエストを完全に解析できないため、リクエストをログに記録できません。リクエストがログ記録されないため、返された HTTP エラーステータスコードをログファイルで表示できません。

リクエストボディが最大サイズを超えると、HTTP エラー状態コードを含むリクエストがログに記録されます。

標準ログの分析

1 時間ごとに複数のアクセスログが配信される可能性があるため、特定の期間に対して受信したすべてのログファイルをまとめて 1 つのファイルにしておくことをお勧めします。これにより、その期間のデータをより正確かつ完全に分析することができます。

アクセスログを分析する方法の 1 つとして Amazon Athena を使用する方法があります。Athena は、CloudFront を含めた AWS のサービスのデータを分析するために役立つインタラクティブなクエリサービスです。詳細については、Amazon Athena ユーザーガイドの「Amazon CloudFront ログのクエリ」を参照してください。

さらに、次の AWS ブログ投稿では、アクセスログを分析するいくつかの方法について説明しています。

- Amazon CloudFront Request Logging (HTTP 経由で配信するコンテンツの場合)
- 強化された CloudFront ログにクエリ文字列機能を追加

> **Important**
>
> ログは、すべてのリクエストを完全に課金するためのものではなく、コンテンツに対するリクエストの本質を把握するものとして使用することをお勧めします。CloudFront はベストエフォートベースでアクセスログを提供します。特定のリクエストのログエントリが、リクエストが実際に処理されてからかなり後に配信されることも、(まれに) 一切配信されないこともあります。ログエントリがアクセスログから省略された場合、アクセスログ内のエントリ数が AWS の利用状況レポートと請求レポートに表示される利用量と一致しなくなります。

標準ログ記録設定の編集

ログ記録の有効化および無効化、ログを保存する Amazon S3 バケットの変更、ログファイルのプレフィックスの変更は、CloudFront コンソールまたは CloudFront API を使用して行うことができます。ログ作成設定の変更は 12 時間以内に有効になります。

詳細については、以下のトピックを参照してください。

- CloudFront コンソールを使用してディストリビューションを更新するには、「ディストリビューションを更新する」を参照してください。
- CloudFront API を使用してディストリビューションを更新する方法については、Amazon CloudFront API リファレンスの「UpdateDistribution」を参照してください。

Amazon S3 バケットからの標準ログファイルの削除

CloudFront では、Amazon S3 バケットからの自動的なログファイル削除は行われません。Amazon S3 バケットからログファイルを削除する方法については、次のトピックを参照してください。

- Amazon S3 コンソールの使用: Amazon Simple Storage Service Console ユーザーガイドの「オブジェクトの削除」。
- REST API の使用: Amazon Simple Storage Service API リファレンスの「DeleteObject」。

標準ログファイル形式

ログファイルには、1 つのビューワーリクエストの詳細が 1 エントリとして記録されます。ログファイルの特性は次のとおりです。

- W3C 拡張ログファイル形式を使用します。

- タブ区切りの値が含まれます。
- レコードが必ずしも時系列順に含まれているとは限りません。
- 2 つのヘッダー行が含まれます。1 つのヘッダー行にファイル形式のバージョンが示され、もう 1 つのヘッダー行に、各レコードに含まれる W3C フィールドが示されます。
- フィールド値に URL エンコードされたスペースおよび特定の他の文字を含めます。

  URL エンコードされた同等の文字は、次の文字に使用されます。

  - ASCII 文字コード 0 ~ 32 以内
  - ASCII 文字コード 127 以上
  - 次の表のすべての文字

  URL エンコーディング標準は RFC 1738 で定義されています。

URL エンコードされた値	文字
%3C	<
%3E	>
%22	"
%23	#
%25	%
%7B	{
%7D	}
%7C	|
%5C	\
%5E	^
%7E	~
%5B	[

URL エンコードされた値	文字
%5D	]
%60	`
%27	'
%20	スペース

標準ログファイルフィールド

ディストリビューションのログファイルには、33 のフィールドが含まれています。次のリストは、各フィールド名と、そのフィールドに保持される情報の説明を順番に示しています。

1. **`date`**

   イベントが発生した日付。YYYY-MM-DD 形式です。例えば、`2019-06-30` と指定します。日付と時刻は協定世界時 (UTC) です。WebSocket 接続の場合、これは接続が閉じた日付です。

2. **`time`**

   CloudFront サーバーがリクエストへの対応を完了した時刻 (UTC) (`01:42:39` など)。WebSocket 接続の場合、これは接続を閉じる時間です。

3. **`x-edge-location`**

   リクエストを処理したエッジロケーション。各エッジロケーションは、3 文字コードと、割り当てられた任意の数字で識別されます (例: DFW3)。通常、この 3 文字コードは、エッジロケーションの地理的場所の近くにある空港の、国際航空運送協会 (IATA) の空港コードに対応します。(これらの略語は今後変更される可能性があります)。

4. **`sc-bytes`**

   サーバーがリクエストに応じてビューワーに送信したデータ (ヘッダーを含む) のバイトの合計数。WebSocket 接続の場合、これは接続を経由してサーバーからクライアントに送信した合計バイト数です。

5. **`c-ip`**

   リクエスト元のビューワーの IP アドレス (`192.0.2.183` または `2001:0db8:85a3::8a2e:0370:7334` など)。ビューワーが HTTP プロキシまたはロードバラ

ンサーを使用してリクエストを送った場合、このフィールドの値はプロキシまたはロードバランサーの IP アドレスです。x-forwarded-for フィールドも参照してください。

6. **cs-method**

   ビューワーから受信した HTTP リクエストメソッド。

7. **cs(Host)**

   CloudFront ディストリビューションのドメイン名 (d111111abcdef8.cloudfront.net など)。

8. **cs-uri-stem**

   パスとオブジェクトを識別するリクエスト URL の部分 (/images/cat.jpg など)。URL 内の疑問符 (?) およびクエリ文字列はログに含まれません。

9. **sc-status**

   次のいずれかの値が含まれます。

   - サーバーのレスポンスの HTTP ステータスコード (例: 200)。
   - 000。この値は、サーバーがリクエストに応答する前に、ビューワーが接続を閉じたことを示します。サーバーがレスポンスの送信を開始した後にビューワーが接続を閉じた場合、このフィールドには、サーバーが送信を開始したレスポンスの HTTP ステータスコードが含まれます。

10. **cs(Referer)**

    リクエスト内の Referer ヘッダーの値。これはリクエスト元のドメインの名前です。一般的なリファラーとして、検索エンジン、オブジェクトに直接リンクされた他のウェブサイト、ユーザー自身のウェブサイトなどがあります。

11. **cs(User-Agent)**

    リクエスト内の User-Agent ヘッダーの値。User-Agent ヘッダーでリクエスト元 (リクエスト元のデバイスとブラウザのタイプなど) が識別されます。リクエスト元が検索エンジンの場合は、どの検索エンジンかも識別されます。

12. **cs-uri-query**

    リクエスト URL のクエリ文字列の部分 (ある場合)。

    URL にクエリ文字列が含まれない場合、このフィールドの値はハイフン (-) です。詳細については、「クエリ文字列パラメータに基づいてコンテンツをキャッシュする」を参照してください。

13. **cs(Cookie)**

名前と値のペアおよび関連属性を含む、リクエスト内の Cookie ヘッダー。

Cookie のログ作成を有効にした場合は、どの Cookie についてオリジンの転送を指定したかに関係なく、CloudFront ではすべてのリクエスト内の Cookie がログに記録されます。リクエストに Cookie ヘッダーが含まれていない場合、このフィールドの値はハイフン (-) です。Cookie の詳細については、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。

14 **x-edge-result-type**

サーバーが、最後のバイトを渡した後で、レスポンスを分類した方法。場合によっては、サーバーがレスポンスを送る準備ができたときから、サーバーがレスポンスを送り終わるまでの間に、結果タイプが変わることがあります。x-edge-response-result-type フィールドも参照してください。

例えば、HTTP ストリーミングで、サーバーがキャッシュ内でストリームのセグメントを検出するとします。そのシナリオでは、このフィールドの値は、通常 Hit になります。この場合、サーバーがセグメント全体を配信する前にビューワーが接続を閉じると、最終結果タイプ (およびこのフィールドの値) は Error になります。

WebSocket 接続の場合、コンテンツがキャッシュ可能ではなく、オリジンに直接プロキシされるため、このフィールドの値は Miss になります。

以下に示しているのは、可能な値です。

- Hit – サーバーがキャッシュからビューワーにオブジェクトを渡しました。
- RefreshHit – サーバーはキャッシュ内でオブジェクトを検出しましたが、オブジェクトの有効期限が切れていたため、サーバーはオリジンに問い合わせて、キャッシュ内に最新バージョンのオブジェクトがあるかどうかを確認しました。
- Miss – キャッシュ内のオブジェクトでリクエストに対応できなかったため、サーバーはリクエストをオリジンに転送して結果をビューワーに返しました。
- LimitExceeded – CloudFront クォータ (以前は制限と呼ばれていました) を超えたため、リクエストは拒否されました。
- CapacityExceeded – リクエストの受信時にサーバーの容量不足でオブジェクトを渡すことができなかったために、サーバーから HTTP 503 ステータスコードが返されました。
- Error – 通常、これはリクエストがクライアントエラーとなった (sc-status フィールドが 4xx 範囲内の値となる)、またはサーバーエラーになった (sc-status フィールドが 5xx 範囲内の値となる) ことを意味します。sc-status フィールドの値が 200 であるか、このフィールドの値が Error で、x-edge-response-result-type フィールドの値が Error でない場合

は、HTTP リクエストは成功したが、クライアントがすべてのバイトを受信する前に切断されたことを意味します。

- Redirect – サーバーは、ディストリビューション設定に従って HTTP から HTTPS にビューワーをリダイレクトしました。

15**x-edge-request-id**

リクエストを一意に識別する不透明な文字列。CloudFront では、この文字列を x-amz-cf-id レスポンスヘッダーでも送信します。

16**x-host-header**

ビューワーが、このリクエストの Host ヘッダーに追加した値。オブジェクトの URL に CloudFront ドメイン名を使用している場合 (d111111abcdef8.cloudfront.net など)、このフィールドにはそのドメイン名が含まれます。代替ドメイン名 (CNAME) をオブジェクト URL (www.example.com) に使用している場合、このフィールドにはその代替ドメイン名が含まれます。

代替ドメイン名を使っている場合には、フィールド 7 の cs(Host) で、ユーザーのディストリビューションに関連するドメイン名を確認します。

17**cs-protocol**

ビューワーリクエストのプロトコル (http、https、ws、wss のいずれか)。

18**cs-bytes**

ビューワーがリクエストに含めたデータ (ヘッダーを含む) のバイトの合計数。WebSocket 接続の場合、これは接続でクライアントからサーバーに送信した合計バイト数です。

19**time-taken**

サーバーが、ビューワーリクエストを受信してからレスポンスの最後のバイトを出力キューに書き込むまでの秒数。サーバーで 1,000 分の 1 秒単位まで測定されます (例: 0.082)。ビューワーから見た場合、レスポンス全体を取得する合計所要時間は、ネットワークのレイテンシーと TCP バッファリングにより、この値よりも長くなります。

20**x-forwarded-for**

ビューワーが HTTP プロキシまたはロードバランサーを使用してリクエストを送信した場合、c-ip フィールドの値はプロキシまたはロードバランサーの IP アドレスです。この場合、このフィールドはリクエスト元のビューワーの IP アドレスです。このフィールドには、複数の IP アドレスをカンマで区切って含めることができます。各 IP アドレスは、IPv4 アドレス

(192.0.2.183 など) または IPv6 アドレス (2001:0db8:85a3::8a2e:0370:7334 など) にすることができます。

ビューワーが HTTP プロキシまたはロードバランサーを使用しなかった場合、このフィールドの値はハイフン (-) です。

21 **ssl-protocol**

リクエストが HTTPS を使用した場合、このフィールドには、リクエストとレスポンスを送信するためにビューワーとサーバーがネゴシエートした SSL/TLS プロトコルが含まれます。指定可能な値のリストについては、ビューワーと CloudFront との間でサポートされているプロトコルと暗号 でサポートされている SSL/TLS プロトコルを参照してください。

フィールド 17 の `cs-protocol` が `http` である場合、このフィールドの値はハイフン (-) です。

22 **ssl-cipher**

リクエストが HTTPS を使用した場合、このフィールドには、リクエストとレスポンスを暗号化するためにビューワーとサーバーがネゴシエートした SSL/TLS 暗号が含まれます。使用できる値のリストについては、「ビューワーと CloudFront との間でサポートされているプロトコルと暗号」で、サポートされている SSL/TLS 暗号化を参照してください。

フィールド 17 の `cs-protocol` が `http` である場合、このフィールドの値はハイフン (-) です。

23 **x-edge-response-result-type**

ビューワーにレスポンスを返す直前にサーバーがレスポンスを分類した方法。`x-edge-result-type` フィールドも参照してください。以下に示しているのは、可能な値です。

- `Hit` – サーバーがキャッシュからビューワーにオブジェクトを渡しました。
- `RefreshHit` – サーバーはキャッシュ内でオブジェクトを検出しましたが、オブジェクトの有効期限が切れていたため、サーバーはオリジンに問い合わせて、キャッシュ内に最新バージョンのオブジェクトがあるかどうかを確認しました。
- `Miss` – キャッシュ内のオブジェクトでリクエストに対応できなかったため、サーバーはリクエストをオリジンサーバーに転送して結果をビューワーに返しました。
- `LimitExceeded` – CloudFront クォータ (以前は制限と呼ばれていました) を超えたため、リクエストは拒否されました。
- `CapacityExceeded` – リクエストの受信時にサーバーの容量不足でオブジェクトを渡すことができなかったために、サーバーから 503 エラーが返されました。

- Error – 通常、これはリクエストがクライアントエラーとなった (sc-status フィールドが 4xx 範囲内の値となる)、またはサーバーエラーになった (sc-status フィールドが 5xx 範囲内の値となる) ことを意味します。

  x-edge-result-type フィールドの値が Error であり、このフィールドの値が Error でない場合、ダウンロードが完了する前にクライアントが切断されました。

- Redirect – サーバーは、ディストリビューション設定に従って HTTP から HTTPS にビューワーをリダイレクトしました。

24.**cs-protocol-version**

ビューワーがリクエストで指定した HTTP バージョン。指定できる値には、HTTP/0.9、HTTP/1.0、HTTP/1.1、HTTP/2.0および HTTP/3.0 などがあります。

25.**fle-status**

フィールドレベル暗号化がディストリビューション用に設定されている場合、このフィールドにはリクエストボディが正常に処理されたかどうかを示すコードが含まれます。サーバーがリクエストボディを正常に処理し、指定したフィールドの値を暗号化してリクエストをオリジンに転送すると、このフィールドの値は Processed になります。x-edge-result-type の値は、この場合でもクライアント側またはサーバー側のエラーを示すことができます。

このフィールドで使用できる値は次のとおりです。

- ForwardedByContentType – コンテンツタイプが設定されていないため、サーバーは解析や暗号化を行わずにリクエストをオリジンに転送しました。
- ForwardedByQueryArgs – フィールドレベル暗号化の設定にないクエリ引数がリクエストに含まれているため、サーバーは解析や暗号化を行わずにリクエストをオリジンに転送しました。
- ForwardedDueToNoProfile – フィールドレベル暗号化の設定でプロファイルを指定しなかったため、サーバーは解析や暗号化を行わずにリクエストをオリジンに転送しました。
- MalformedContentTypeClientError Content-Type – ヘッダーの値が無効な形式であるため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- MalformedInputClientError – リクエストボディが無効な形式であるため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- MalformedQueryArgsClientError – クエリ引数が空であるか無効な形式であるため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。

- RejectedByContentType – フィールドレベル暗号化の設定でコンテンツタイプを指定しなかったため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- RejectedByQueryArgs – フィールドレベル暗号化の設定でクエリ引数を指定しなかったため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- ServerError – オリジンサーバーがエラーを返しました。

リクエストがフィールドレベル暗号化のクォータ (以前は制限と呼ばれていました) を超えた場合、このフィールドには次のいずれかのエラーコードが含まれ、サーバーは HTTP ステータスコード 400 をビューワーに返します。フィールドレベル暗号化に関する最新のクォータのリストについては、「フィールドレベル暗号化のクォータ」を参照してください。

- FieldLengthLimitClientError – 暗号化されるように設定されているフィールドが最大の長さを超えています。
- FieldNumberLimitClientError – ディストリビューションによって暗号化されるように設定されているリクエストがフィールド数の制限を超えています。
- RequestLengthLimitClientError – フィールドレベル暗号化が設定されている場合にリクエストボディが最大の長さを超えています。

フィールドレベル暗号化がディストリビューション用に設定されていない場合、このフィールドの値はハイフン (-) です。

26. **fle-encrypted-fields**

サーバーが暗号化してオリジンに転送したフィールドレベル暗号化フィールドの数。CloudFront サーバーは処理されたリクエストをオリジンにストリーミングするときにデータを暗号化するため、fle-status の値がエラーであっても、このフィールドに値が渡されている場合があります。

フィールドレベル暗号化がディストリビューション用に設定されていない場合、このフィールドの値はハイフン (-) です。

27. **c-port**

閲覧者からのリクエストのポート番号。

28. **time-to-first-byte**

サーバー上で測定される、要求を受信してから応答の最初のバイトを書き込むまでの秒数。

29**x-edge-detailed-result-type**

このフィールドには、以下の場合を除き、x-edge-result-type フィールドと同じ値が含まれます。

- オブジェクトが Origin Shield レイヤーからビューワーに渡された場合、このフィールドには OriginShieldHit が含まれています。
- オブジェクトが CloudFront キャッシュに存在せず、レスポンスがオリジンリクエストの Lambda@Edge 関数によって生成された場合、このフィールドには MissGeneratedResponse が含まれます。
- x-edge-result-type フィールドの値が Error の場合、このフィールドにはエラーに関する詳細情報を含む次のいずれかの値が含まれます。
  - AbortedOrigin – サーバーでオリジンに関する問題が発生しました。
  - ClientCommError – サーバーとビューワーとの通信の問題により、ビューワーへのレスポンスが中断されました。
  - ClientGeoBlocked – ディストリビューションは、ビューワーの地理的位置からのリクエストを拒否するように設定されています。
  - ClientHungUpRequest – リクエストの送信中にビューワーが途中で停止しました。
  - Error – エラータイプが他のどのカテゴリにも適合しないエラーが発生しました。このエラータイプは、キャッシュからのエラーレスポンスをサーバーが渡すときに発生する可能性があります。
  - InvalidRequest – サーバーがビューワーから無効なリクエストを受信しました。
  - InvalidRequestBlocked – 要求されたリソースへのアクセスがブロックされます。
  - InvalidRequestCertificate – ディストリビューションが、HTTPS 接続の確立に使用した SSL/TLS 証明書と一致しません。
  - InvalidRequestHeader – リクエストに無効なヘッダーが含まれていました。
  - InvalidRequestMethod – ディストリビューションは、使用された HTTP リクエストメソッドを処理するように設定されていません。これは、ディストリビューションがキャッシュ可能なリクエストのみをサポートしている場合に発生します。
  - OriginCommError — オリジンに接続中、またはオリジンからデータを読み取るときに、リクエストがタイムアウトしました。
  - OriginConnectError – サーバーがオリジンに接続できませんでした。
  - OriginContentRangeLengthError – オリジンのレスポンスの Content-Length ヘッダーが、Content-Range ヘッダーの長さと一致しません。

- OriginDnsError – サーバーがオリジンのドメイン名を解決できませんでした。
- OriginError – オリジンが誤ったレスポンスを返しました。
- OriginHeaderTooBigError – オリジンから返されたヘッダーが大きすぎてエッジサーバーで処理できません。
- OriginInvalidResponseError – オリジンが無効なレスポンスを返しました。
- OriginReadError – サーバーがオリジンから読み取れませんでした。
- OriginWriteError – サーバーがオリジンに書き込めませんでした。
- OriginZeroSizeObjectError – オリジンから送信されたサイズゼロのオブジェクトがエラーになりました。
- SlowReaderOriginError – オリジンエラーの原因となったメッセージの読み取りに時間がかかりました。

30**sc-content-type**

レスポンスの HTTP Content-Type ヘッダーの値。

31**sc-content-len**

レスポンスの HTTP Content-Length ヘッダーの値。

32**sc-range-start**

レスポンスに HTTP Content-Range ヘッダーが含まれている場合、このフィールドには範囲の開始値が含まれます。

33**sc-range-end**

レスポンスに HTTP Content-Range ヘッダーが含まれている場合、このフィールドには範囲の終了値が含まれます。

ディストリビューションのログファイルの例を以下に示します。

```
#Version: 1.0
#Fields: date time x-edge-location sc-bytes c-ip cs-method cs(Host) cs-uri-stem sc-
status cs(Referer) cs(User-Agent) cs-uri-query cs(Cookie) x-edge-result-type x-edge-
request-id x-host-header cs-protocol cs-bytes time-taken x-forwarded-for ssl-protocol
 ssl-cipher x-edge-response-result-type cs-protocol-version fle-status fle-encrypted-
fields c-port time-to-first-byte x-edge-detailed-result-type sc-content-type sc-
content-len sc-range-start sc-range-end
2019-12-04 21:02:31 LAX1 392 192.0.2.100 GET d111111abcdef8.cloudfront.net /
index.html 200 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;
```

```
%20x64)%20AppleWebKit/537.36%20(KHTML,%20like
%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
 SOX4xwn4XV6Q4rgb7XiVGOHms_BGlTAC4KyHmureZmBNrjGdRLiNIQ== d111111abcdef8.cloudfront.net
 https 23 0.001 - TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 Hit HTTP/2.0 - - 11040 0.001 Hit
 text/html 78 - -
2019-12-04 21:02:31 LAX1 392 192.0.2.100 GET d111111abcdef8.cloudfront.net /
index.html 200 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;
%20x64)%20AppleWebKit/537.36%20(KHTML,%20like
%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
 k6WGMNkEzR5BEM_SaF47gjtX9zBDO2m349OY2an0QPEaUum1ZOLrow== d111111abcdef8.cloudfront.net
 https 23 0.000 - TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 Hit HTTP/2.0 - - 11040 0.000 Hit
 text/html 78 - -
2019-12-04 21:02:31 LAX1 392 192.0.2.100 GET d111111abcdef8.cloudfront.net /
index.html 200 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;
%20x64)%20AppleWebKit/537.36%20(KHTML,%20like
%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
 f37nTMVvnKvV2ZSvEsivup_c2kZ7VXzYdjC-GUQZ5qNs-89BlWazbw== d111111abcdef8.cloudfront.net
 https 23 0.001 - TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 Hit HTTP/2.0 - - 11040 0.001 Hit
 text/html 78 - -
2019-12-13 22:36:27 SEA19-C1 900 192.0.2.200 GET d111111abcdef8.cloudfront.net /
favicon.ico 502 http://www.example.com/ Mozilla/5.0%20(Windows
%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Error
 1pkpNfBQ39sYMnjjUQjmH2w1wdJnbHYTbag21o_3OfcQgPzdL2RSSQ== www.example.com http 675
 0.102 - - - Error HTTP/1.1 - - 25260 0.102 OriginDnsError text/html 507 - -
2019-12-13 22:36:26 SEA19-C1 900 192.0.2.200 GET d111111abcdef8.cloudfront.net / 502
 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Error
 3AqrZGCnF_g0-5KOvfA7c9XLcf4YGvMFSeFdIetR1N_2y8jSis8Zxg== www.example.com http 735
 0.107 - - - Error HTTP/1.1 - - 3802 0.107 OriginDnsError text/html 507 - -
2019-12-13 22:37:02 SEA19-C2 900 192.0.2.200 GET d111111abcdef8.cloudfront.net / 502
 - curl/7.55.1 - - Error kBkDzGnceVtWHqSCqBUqtA_cEs2T3tFUBbnBNkB9El_uVRhHgcZfcw==
 www.example.com http 387 0.103 - - - Error HTTP/1.1 - - 12644 0.103 OriginDnsError
 text/html 507 - -
```

標準ログの料金

標準ログ記録は、CloudFront のオプション機能です。標準ログ記録を有効にしても追加料金はかかりません。ただし、Amazon S3 でのファイルの保存とアクセス用に通常の Amazon S3 料金が発生します (ファイルはいつでも削除できます)。

Amazon S3 の料金に関する詳細については、「Amazon S3 の料金」を参照してください。

CloudFront の料金の詳細については、「CloudFront の料金」を参照してください。

リアルタイムログ

CloudFront リアルタイムログを使用すると、ディストリビューションに対するリクエストの情報をリアルタイムで取得できます (ログはリクエストを受信してから数秒以内に配信されます)。リアルタイムログを使用して、コンテンツ配信のパフォーマンスに基づいて監視、分析、アクションを実行できます。

CloudFront リアルタイムログは設定可能です。以下を選択することができます。

- リアルタイムログのサンプリング率 (リアルタイムのログ記録を受信するリクエストの割合) を選択できます。
- ログレコードで受信する特定のフィールド。
- リアルタイムログを受信する特定のキャッシュ動作 (パスパターン)。

CloudFront リアルタイムログは、Amazon Kinesis Data Streams で選択したデータストリームに配信されます。独自の Kinesis データストリームコンシューマーを構築することも、Amazon Data Firehose を使用してログデータを Amazon Simple Storage Service (Amazon S3)、Amazon Redshift、Amazon OpenSearch Service (OpenSearch Service)、またはサードパーティーのログ処理サービスに送信することもできます。

Kinesis Data Streams の使用料金に加えて、CloudFront でのリアルタイムログの料金が発生します。料金の詳細については、「Amazon CloudFront の料金」および「Amazon Kinesis Data Streams の料金」を参照してください。

> **Important**
>
> ログは、すべてのリクエストを完全に課金するためのものではなく、コンテンツに対するリクエストの本質を把握するものとして使用することをお勧めします。CloudFront はベストエフォートベースでリアルタイムのアクセスログを提供します。特定のリクエストのログエントリが、リクエストが実際に処理されてからかなり後に配信されることも、(まれに) 一切配信されないこともあります。ログエントリをリアルタイムログから省略すると、リアルタイムログ内のエントリ数は AWS の請求と使用状況レポートに表示される使用量と一致しなくなります。

リアルタイムログ設定について

CloudFront リアルタイムログを使用するには、まずリアルタイムログ設定を作成します。リアルタイムログ設定には、受信するログフィールド、ログレコードの サンプリングレート、およびログを配信する Kinesis データストリームに関する情報が含まれます。

具体的には、リアルタイムログ設定には、次の設定が含まれます。

- 名前
- サンプリングレート
- フィールド
- エンドポイント (Kinesis データストリーム)
- IAM ロール

名前

リアルタイムログ設定を識別する名前。

サンプリングレート

サンプリングレートは、リアルタイムログレコードとして Kinesis Data Streams に送信されるビューワーリクエストの割合を決定する 1 ~ 100 の整数です。すべてのビューワーリクエストをリアルタイムログに含めるには、サンプリングレートに 100 を指定します。リクエストデータの代表的なサンプルをリアルタイムログに受信しながら、コストを削減するために、より低いサンプリングレートを選択することもできます。

フィールド

各リアルタイムログレコードに含まれるフィールドのリスト。各ログレコードには、最大 40 個のフィールドを含めることができます。使用可能なすべてのフィールドを受信するか、パフォーマンスのモニタリングと分析に必要なフィールドのみを受信するかを選択できます。

次のリストは、各フィールド名と、そのフィールドに保持される情報の説明を示しています。フィールドは、Kinesis Data Streams に配信されるログレコードに表示される順序で示されています。

フィールド 46～63 は、メディアプレーヤークライアントが各リクエストで CDN に送信できる一般的なメディアクライアントデータ (CMCD) です。このデータを使用して、メディアタイプ (オーディオ、動画)、再生速度、ストリーミング時間など、各リクエストを確認できます。これらのフィールドは、CloudFront に送信した場合にのみ、リアルタイムログに表示されます。

1. **timestamp**

   エッジサーバーがリクエストへの応答を終了した日時。

2. **c-ip**

   リクエスト元のビューワーの IP アドレス (192.0.2.183 または 2001:0db8:85a3::8a2e:0370:7334 など)。ビューワーが HTTP プロキシまたはロードバランサーを使用してリクエストを送った場合、このフィールドの値はプロキシまたはロードバランサーの IP アドレスです。x-forwarded-for フィールドも参照してください。

3. **time-to-first-byte**

   サーバー上で測定される、要求を受信してから応答の最初のバイトを書き込むまでの秒数。

4. **sc-status**

   サーバーのレスポンスの HTTP ステータスコード (例: 200)。

5. **sc-bytes**

   サーバーがリクエストに応じてビューワーに送信したデータ (ヘッダーを含む) のバイトの合計数。WebSocket 接続の場合、これは接続を経由してサーバーからクライアントに送信した合計バイト数です。

6. **cs-method**

   ビューワーから受信した HTTP リクエストメソッド。

7. **cs-protocol**

   ビューワーリクエストのプロトコル (http、https、ws、wss のいずれか)。

8. **cs-host**

   ビューワーが、このリクエストの Host ヘッダーに追加した値。オブジェクトの URL に CloudFront ドメイン名を使用している場合 (d111111abcdef8.cloudfront.net など)、このフィールドにはそのドメイン名が含まれます。代替ドメイン名 (CNAME) をオブジェクト URL (www.example.com) に使用している場合、このフィールドにはその代替ドメイン名が含まれます。

9. **cs-uri-stem**

   クエリ文字列 (存在する場合) を含むが、ドメイン名を含まないリクエスト URL 全体。例えば、/images/cat.jpg?mobile=true と指定します。

> **Note**
>
> 標準ログでは、cs-uri-stem 値にクエリ文字列は含まれません。

10. **cs-bytes**

    ビューワーがリクエストに含めたデータ (ヘッダーを含む) のバイトの合計数。WebSocket 接続の場合、これは接続でクライアントからサーバーに送信した合計バイト数です。

11. **x-edge-location**

    リクエストを処理したエッジロケーション。各エッジロケーションは、3 文字コードと、割り当てられた任意の数字で識別されます (例: DFW3)。通常、この 3 文字コードは、エッジロケーションの地理的場所の近くにある空港の、国際航空運送協会 (IATA) の空港コードに対応します。(これらの略語は今後変更される可能性があります)。

12. **x-edge-request-id**

    リクエストを一意に識別する不透明な文字列。CloudFront では、この文字列を x-amz-cf-id レスポンスヘッダーでも送信します。

13. **x-host-header**

    CloudFront ディストリビューションのドメイン名 (d111111abcdef8.cloudfront.net など)。

14. **time-taken**

    サーバーが、ビューワーのリクエストを受信してからレスポンスの最後のバイトを出力キューに書き込むまでの秒数。サーバーで 1,000 分の 1 秒単位まで測定されます (例: 0.082)。ビューワーから見た場合、レスポンス全体を取得する合計所要時間は、ネットワークのレイテンシーと TCP バッファリングにより、この値よりも長くなります。

15. **cs-protocol-version**

    ビューワーがリクエストで指定した HTTP バージョン。指定できる値には、HTTP/0.9、HTTP/1.0、HTTP/1.1、HTTP/2.0および HTTP/3.0 などがあります。

16. **c-ip-version**

    リクエストの IP バージョン (IPv4 または IPv6)。

17. **cs-user-agent**

リクエスト内の User-Agent ヘッダーの値。User-Agent ヘッダーでリクエスト元 (リクエスト元のデバイスとブラウザのタイプなど) が識別されます。リクエスト元が検索エンジンの場合は、どの検索エンジンかも識別されます。

18. **cs-referer**

リクエスト内の Referer ヘッダーの値。これはリクエスト元のドメインの名前です。一般的なリファラーとして、検索エンジン、オブジェクトに直接リンクされた他のウェブサイト、ユーザー自身のウェブサイトなどがあります。

19. **cs-cookie**

名前と値のペアおよび関連属性を含む、リクエスト内の Cookie ヘッダー。

> **Note**
>
> このフィールドは 800 バイトに切り捨てられます。

20. **cs-uri-query**

リクエスト URL のクエリ文字列の部分 (ある場合)。

21. **x-edge-response-result-type**

ビューワーにレスポンスを返す直前にサーバーがレスポンスを分類した方法。x-edge-result-type フィールドも参照してください。以下に示しているのは、可能な値です。

- Hit – サーバーがキャッシュからビューワーにオブジェクトを渡しました。
- RefreshHit – サーバーはキャッシュ内でオブジェクトを検出しましたが、オブジェクトの有効期限が切れていたため、サーバーはオリジンに問い合わせて、キャッシュ内に最新バージョンのオブジェクトがあるかどうかを確認しました。
- Miss – キャッシュ内のオブジェクトでリクエストに対応できなかったため、サーバーはリクエストをオリジンサーバーに転送して結果をビューワーに返しました。
- LimitExceeded – CloudFront クォータ (以前は制限と呼ばれていました) を超えたため、リクエストは拒否されました。
- CapacityExceeded – リクエストの受信時にサーバーの容量不足でオブジェクトを渡すことができなかったために、サーバーから 503 エラーが返されました。
- Error – 通常、これはリクエストがクライアントエラーとなった (sc-status フィールドが 4xx 範囲内の値となる)、またはサーバーエラーになった (sc-status フィールドが 5xx 範囲内の値となる) ことを意味します。

x-edge-result-type フィールドの値が Error であり、このフィールドの値が Error でない場合、ダウンロードが完了する前にクライアントが切断されました。

- Redirect – サーバーは、ディストリビューション設定に従って HTTP から HTTPS にビューワーをリダイレクトしました。

22 **x-forwarded-for**

ビューワーが HTTP プロキシまたはロードバランサーを使用してリクエストを送信した場合、c-ip フィールドの値はプロキシまたはロードバランサーの IP アドレスです。この場合、このフィールドはリクエスト元のビューワーの IP アドレスです。このフィールドには、複数の IP アドレスをカンマで区切って含めることができます。各 IP アドレスは、IPv4 アドレス (192.0.2.183 など) または IPv6 アドレス (2001:0db8:85a3::8a2e:0370:7334 など) にすることができます。

23 **ssl-protocol**

リクエストが HTTPS を使用した場合、このフィールドには、リクエストとレスポンスを送信するためにビューワーとサーバーがネゴシエートした SSL/TLS プロトコルが含まれます。指定可能な値のリストについては、ビューワーと CloudFront との間でサポートされているプロトコルと暗号 でサポートされている SSL/TLS プロトコルを参照してください。

24 **ssl-cipher**

リクエストが HTTPS を使用した場合、このフィールドには、リクエストとレスポンスを暗号化するためにビューワーとサーバーがネゴシエートした SSL/TLS 暗号が含まれます。使用できる値のリストについては、「ビューワーと CloudFront との間でサポートされているプロトコルと暗号」で、サポートされている SSL/TLS 暗号化を参照してください。

25 **x-edge-result-type**

サーバーが、最後のバイトを渡した後で、レスポンスを分類した方法。場合によっては、サーバーがレスポンスを送る準備ができたときから、サーバーがレスポンスを送り終わるまでの間に、結果タイプが変わることがあります。x-edge-response-result-type フィールドも参照してください。

例えば、HTTP ストリーミングで、サーバーがキャッシュ内でストリームのセグメントを検出するとします。そのシナリオでは、このフィールドの値は、通常 Hit になります。この場合、サーバーがセグメント全体を配信する前にビューワーが接続を閉じると、最終結果タイプ (およびこのフィールドの値) は Error になります。

WebSocket 接続の場合、コンテンツがキャッシュ可能ではなく、オリジンに直接プロキシされるため、このフィールドの値は Miss になります。

以下に示しているのは、可能な値です。

- Hit – サーバーがキャッシュからビューワーにオブジェクトを渡しました。
- RefreshHit – サーバーはキャッシュ内でオブジェクトを検出しましたが、オブジェクトの有効期限が切れていたため、サーバーはオリジンに問い合わせて、キャッシュ内に最新バージョンのオブジェクトがあるかどうかを確認しました。
- Miss – キャッシュ内のオブジェクトでリクエストに対応できなかったため、サーバーはリクエストをオリジンに転送して結果をビューワーに返しました。
- LimitExceeded – CloudFront クォータ (以前は制限と呼ばれていました) を超えたため、リクエストは拒否されました。
- CapacityExceeded – リクエストの受信時にサーバーの容量不足でオブジェクトを渡すことができなかったために、サーバーから HTTP 503 ステータスコードが返されました。
- Error – 通常、これはリクエストがクライアントエラーとなった (sc-status フィールドが 4xx 範囲内の値となる)、またはサーバーエラーになった (sc-status フィールドが 5xx 範囲内の値となる) ことを意味します。sc-status フィールドの値が 200 であるか、このフィールドの値が Error で、x-edge-response-result-type フィールドの値が Error でない場合は、HTTP リクエストは成功したが、クライアントがすべてのバイトを受信する前に切断されたことを意味します。
- Redirect – サーバーは、ディストリビューション設定に従って HTTP から HTTPS にビューワーをリダイレクトしました。

26. **fle-encrypted-fields**

サーバーが暗号化してオリジンに転送したフィールドレベル暗号化フィールドの数。CloudFront サーバーは処理されたリクエストをオリジンにストリーミングするときにデータを暗号化するため、fle-status の値がエラーであっても、このフィールドに値が渡されている場合があります。

27. **fle-status**

フィールドレベル暗号化がディストリビューション用に設定されている場合、このフィールドにはリクエストボディが正常に処理されたかどうかを示すコードが含まれます。サーバーがリクエストボディを正常に処理し、指定したフィールドの値を暗号化してリクエストをオリジンに転送すると、このフィールドの値は Processed になります。x-edge-result-type の値は、この場合でもクライアント側またはサーバー側のエラーを示すことができます。

このフィールドで使用できる値は次のとおりです。

- ForwardedByContentType – コンテンツタイプが設定されていないため、サーバーは解析や暗号化を行わずにリクエストをオリジンに転送しました。
- ForwardedByQueryArgs – フィールドレベル暗号化の設定にないクエリ引数がリクエストに含まれているため、サーバーは解析や暗号化を行わずにリクエストをオリジンに転送しました。
- ForwardedDueToNoProfile – フィールドレベル暗号化の設定でプロファイルを指定しなかったため、サーバーは解析や暗号化を行わずにリクエストをオリジンに転送しました。
- MalformedContentTypeClientError Content-Type – ヘッダーの値が無効な形式であるため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- MalformedInputClientError – リクエストボディが無効な形式であるため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- MalformedQueryArgsClientError – クエリ引数が空であるか無効な形式であるため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- RejectedByContentType – フィールドレベル暗号化の設定でコンテンツタイプを指定しなかったため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- RejectedByQueryArgs – フィールドレベル暗号化の設定でクエリ引数を指定しなかったため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- ServerError – オリジンサーバーがエラーを返しました。

リクエストがフィールドレベル暗号化のクォータ (以前は制限と呼ばれていました) を超えた場合、このフィールドには次のいずれかのエラーコードが含まれ、サーバーは HTTP ステータスコード 400 をビューワーに返します。フィールドレベル暗号化に関する最新のクォータのリストについては、「フィールドレベル暗号化のクォータ」を参照してください。

- FieldLengthLimitClientError – 暗号化されるように設定されているフィールドが最大の長さを超えています。
- FieldNumberLimitClientError – ディストリビューションによって暗号化されるように設定されているリクエストがフィールド数の制限を超えています。
- RequestLengthLimitClientError – フィールドレベル暗号化が設定されている場合にリクエストボディが最大の長さを超えています。

28**sc-content-type**

レスポンスの HTTP Content-Type ヘッダーの値。

29**sc-content-len**

レスポンスの HTTP Content-Length ヘッダーの値。

30**sc-range-start**

レスポンスに HTTP Content-Range ヘッダーが含まれている場合、このフィールドには範囲の開始値が含まれます。

31**sc-range-end**

レスポンスに HTTP Content-Range ヘッダーが含まれている場合、このフィールドには範囲の終了値が含まれます。

32**c-port**

閲覧者からのリクエストのポート番号。

33**x-edge-detailed-result-type**

このフィールドには、以下の場合を除き、x-edge-result-type フィールドと同じ値が含まれます。

- オブジェクトが Origin Shield レイヤーからビューワーに渡された場合、このフィールドには OriginShieldHit が含まれています。
- オブジェクトが CloudFront キャッシュに存在せず、レスポンスがオリジンリクエストの Lambda@Edge 関数によって生成された場合、このフィールドには MissGeneratedResponse が含まれます。
- x-edge-result-type フィールドの値が Error の場合、このフィールドにはエラーに関する詳細情報を含む次のいずれかの値が含まれます。
  - AbortedOrigin – サーバーでオリジンに関する問題が発生しました。
  - ClientCommError – サーバーとビューワーとの通信の問題により、ビューワーへのレスポンスが中断されました。
  - ClientGeoBlocked – ディストリビューションは、ビューワーの地理的位置からのリクエストを拒否するように設定されています。
  - ClientHungUpRequest – リクエストの送信中にビューワーが途中で停止しました。

- Error – エラータイプが他のどのカテゴリにも適合しないエラーが発生しました。このエラータイプは、キャッシュからのエラーレスポンスをサーバーが渡すときに発生する可能性があります。
- InvalidRequest – サーバーがビューワーから無効なリクエストを受信しました。
- InvalidRequestBlocked – 要求されたリソースへのアクセスがブロックされます。
- InvalidRequestCertificate – ディストリビューションが、HTTPS 接続の確立に使用した SSL/TLS 証明書と一致しません。
- InvalidRequestHeader – リクエストに無効なヘッダーが含まれていました。
- InvalidRequestMethod – ディストリビューションは、使用された HTTP リクエストメソッドを処理するように設定されていません。これは、ディストリビューションがキャッシュ可能なリクエストのみをサポートしている場合に発生します。
- OriginCommError — オリジンに接続中、またはオリジンからデータを読み取るときに、リクエストがタイムアウトしました。
- OriginConnectError – サーバーがオリジンに接続できませんでした。
- OriginContentRangeLengthError – オリジンのレスポンスの Content-Length ヘッダーが、Content-Range ヘッダーの長さと一致しません。
- OriginDnsError – サーバーがオリジンのドメイン名を解決できませんでした。
- OriginError – オリジンが誤ったレスポンスを返しました。
- OriginHeaderTooBigError – オリジンから返されたヘッダーが大きすぎてエッジサーバーで処理できません。
- OriginInvalidResponseError – オリジンが無効なレスポンスを返しました。
- OriginReadError – サーバーがオリジンから読み取れませんでした。
- OriginWriteError – サーバーがオリジンに書き込めませんでした。
- OriginZeroSizeObjectError – オリジンから送信されたサイズゼロのオブジェクトがエラーになりました。
- SlowReaderOriginError – オリジンエラーの原因となったメッセージの読み取りに時間がかかりました。

34.**c-country**

ビューワーの IP アドレスによって決定される、ビューワーの地理的位置を表す国コード。国コードの一覧については、「ISO 3166-1 alpha-2」を参照してください。

35.**cs-accept-encoding**

ビューワーリクエスト内の Accept-Encoding ヘッダーの値。

36 **cs-accept**

ビューワーリクエスト内の Accept ヘッダーの値。

37 **cache-behavior-path-pattern**

ビューワーリクエストに一致したキャッシュ動作を識別するパスパターン。

38 **cs-headers**

ビューワーリクエスト内の HTTP ヘッダー (名前と値)。

> **Note**
>
> このフィールドは 800 バイトに切り捨てられます。

39 **cs-header-names**

ビューワーリクエスト内の HTTP ヘッダーの名前 (値ではない)。

> **Note**
>
> このフィールドは 800 バイトに切り捨てられます。

40 **cs-headers-count**

ビューワーリクエスト内の HTTP ヘッダーの数。

41 **origin-fbl**

CloudFront とオリジン間の先頭バイトのレイテンシーの秒数。

42 **origin-lbl**

CloudFront とオリジン間の最終バイトのレイテンシーの秒数。

43 **asn**

ビューワーの AS 番号 (ASN)。

44 **primary-distribution-id**

継続的デプロイが有効になっている場合、この ID は、どのディストリビューションが現在のディストリビューションでプライマリであるかを識別します。

45 **primary-distribution-dns-name**

継続的デプロイが有効になっている場合、この値は、現在の CloudFront ディストリビューションに関連するプライマリドメイン名 (d111111abcdef8.cloudfront.net など) を示します。

> **リアルタイムログの CMCD フィールド**
>
> これらのフィールドの詳細については、「CTA Specification Web Application Video Ecosystem - Common Media Client Data CTA-5004」ドキュメントを参照してください。

46 **cmcd-encoded-bitrate**

リクエストされたオーディオまたは動画オブジェクトのエンコードされたビットレート。

47 **cmcd-buffer-length**

リクエストされたメディアオブジェクトのバッファ長。

48 **cmcd-buffer-starvation**

前のリクエストからオブジェクトリクエストまでの特定の時点でバッファが不足していたかどうか。これにより、プレーヤーが再バッファリング状態になり、動画またはオーディオの再生が停止する可能性があります。

49 **cmcd-content-id**

現在のコンテンツを識別する一意の文字列。

50 **cmcd-object-duration**

リクエストされたオブジェクトの再生時間 (ミリ秒単位)。

51 **cmcd-deadline**

バッファアンダーラン状態やその他の再生問題を回避するために、このオブジェクトの最初のサンプルが利用可能でなければならない、リクエスト時点からの期限。

52 **cmcd-measured-throughput**

クライアントが測定した、クライアントとサーバー間のスループット。

53 **cmcd-next-object-request**

次のリクエストされたオブジェクトの相対パス。

54. **cmcd-next-range-request**

次のリクエストが部分的なオブジェクトリクエストである場合、この文字列はリクエスト対象のバイト範囲を示します。

55. **cmcd-object-type**

リクエスト対象の現在のオブジェクトのメディアタイプ。

56. **cmcd-playback-rate**

リアルタイムの場合は 1、倍速の場合は 2、再生しない場合は 0。

57. **cmcd-requested-maximum-throughput**

クライアントがアセットの配信に十分であると判断した、リクエストされた最大スループット。

58. **cmcd-streaming-format**

現在のリクエストを定義するストリーミングフォーマット。

59. **cmcd-session-id**

現在の再生セッションを識別する GUID。

60. **cmcd-stream-type**

セグメントの可用性を識別するトークン。v = すべてのセグメントが使用可能です。l = セグメントは時間が経つと使用可能になります。

61. **cmcd-startup**

起動、シーク、またはバッファが空になった後の回復中にオブジェクトが緊急に必要になった場合は、値なしでキーが含まれます。

62. **cmcd-top-bitrate**

クライアントが再生できる最高ビットレートのレンディション。

63. **cmcd-version**

定義されたキーの名前と値を解釈するために使用する、この仕様のバージョン。このキーを省略した場合、クライアントとサーバーは、値がバージョン 1 で定義されたものとして解釈する必要があります。

エンドポイント (Kinesis データストリーム)

エンドポイントには、リアルタイムログを送信する Kinesis データストリームに関する情報が含まれています。データストリームの Amazon リソースネーム (ARN) を指定します。

Kinesis データストリームの作成の詳細については、Amazon Kinesis Data Streams 開発者ガイドの以下のトピックを参照してください。

- コンソールを使用したストリームの管理
- AWS CLI を使用した基本的な Kinesis Data Stream オペレーションの実行
- ストリームの作成 (AWS SDK for Java を使用)

データストリームを作成するときは、シャードの数を指定する必要があります。次の情報を使用して、必要なシャードの数を見積もることができます。

Kinesis データストリームのシャード数を推定するには

1. ご使用の CloudFront ディストリビューションが受信する 1 秒あたりのリクエスト数を計算 (または概算) します。

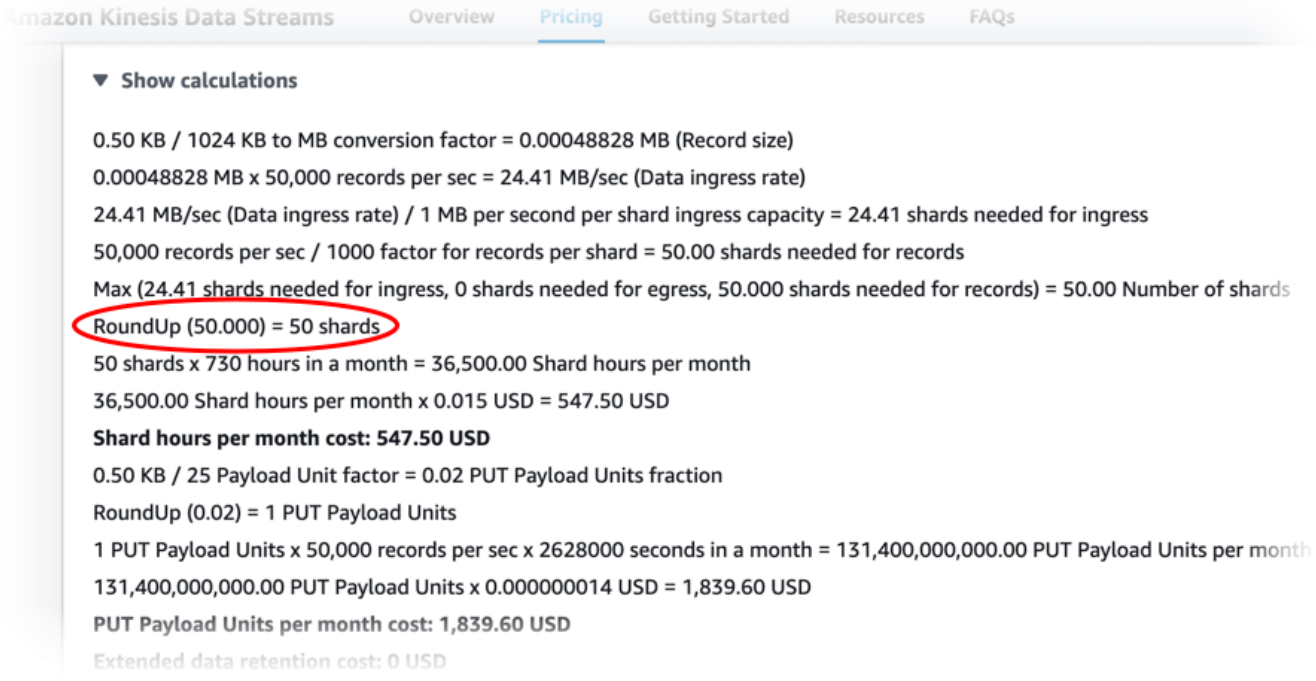
   1 秒あたりのリクエストを計算するには、CloudFront の使用状況レポート (CloudFront コンソール内) と CloudFront メトリクス (CloudFront コンソールおよび Amazon CloudWatch コンソール内) を使用します。
2. 1 つのリアルタイムログレコードの一般的なサイズを決定します。

   一般に、1 つのログレコードは約 500 バイトです。使用可能なすべてのフィールドを含む大きなレコードは、通常、約 1 KB です。

   ログレコードのサイズが不明な場合は、サンプルレートを低く (1% などに) 設定して、リアルタイムログを有効化し、Kinesis Data Streams でのデータのモニタリングを使用して平均的なレコードサイズを割り出します (受信バイト数の合計をレコード数の合計で割ります)。
3. Amazon Kinesis Data Streams の料金のページの [Pricing calculator] (料金見積りツール) で、1 秒あたりのリクエスト (レコード) の数と 1 つのログレコードの平均レコードサイズを入力します。その後、[Show calculations] (計算を表示) を選択します。

   料金見積りツールには、必要なシャードの数が表示されます。(見積もりコストも表示されます。)

   次の例は、平均レコードサイズが 0.5 KB、1 秒あたり 50,000 リクエストの場合、50 個のシャードが必要であることを示しています。



IAM ロール

Kinesis のデータストリームにリアルタイムログを配信するための許可を CloudFront に付与する AWS Identity and Access Management (IAM) ロールです。

CloudFront コンソールでリアルタイムログ設定を作成する場合、[新規サービスロールの作成] を選択して、コンソールで IAM ロールを作成させることができます。

AWS CloudFormation または CloudFront API (AWS CLI もしくは SDK) を使用してリアルタイムログ設定を作成する場合は、独自に IAM ロールを作成して、ロール ARN を提供する必要があります。IAM ロールを自分で作成するには、次のポリシーを使用します。

IAM ロール信頼ポリシー

次の IAM ロール信頼ポリシーを使用するには、*111122223333* を AWS アカウント 数字に置き換えます。このポリシーの `Condition` 要素は、CloudFront が AWS アカウント 内のディストリビューションの代理としてのみこのロールを引き受けることができるため、「混乱した代理」問題を防止するのに役立ちます。

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

```
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                }
            }
        }
    ]
}
```

暗号化されていないデータストリーム用の IAM ロールアクセス許可ポリシー

次のポリシーを使用するには、*arn:aws:kinesis:us-east-2:123456789012:stream/StreamName* を Kinesis Data Streams の ARN に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamSummary",
                "kinesis:DescribeStream",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:us-east-2:123456789012:stream/StreamName"
            ]
        }
    ]
}
```

暗号化されたデータストリーム用の IAM ロールアクセス許可ポリシー

次のポリシーを使用するには、*arn:aws:kinesis:us-east-2:123456789012:stream/StreamName* を Kinesis Data Streams の ARN に、*arn:aws:kms:us-*

east-2:123456789012:key/e58a3d0b-fe4f-4047-a495-ae03cc73d486 を AWS KMS key の ARN に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamSummary",
                "kinesis:DescribeStream",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:us-east-2:123456789012:stream/StreamName"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-2:123456789012:key/e58a3d0b-fe4f-4047-a495-
ae03cc73d486"
            ]
        }
    ]
}
```

リアルタイムログ設定の作成と使用

リアルタイムログ設定を使用してディストリビューションに対して行われたリクエストに関する情報をリアルタイムで取得できます (ログはリクエストを受信してから数秒以内に配信されます)。リアルタイムログ設定は、CloudFront コンソール、AWS Command Line Interface (AWS CLI)、または CloudFront API を使用して作成できます。

リアルタイムログ設定を使用するには、CloudFront ディストリビューションの 1 つ以上のキャッシュ動作にアタッチします。

リアルタイムログ設定の作成 (コンソール)

リアルタイムログ設定を作成するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home?#/logs で CloudFront コンソールの [Logs] (ログ) ページを開きます。
2. [リアルタイム設定] タブを選択します。
3. [Create configuration] (設定を作成) をクリックします。
4. [名前] に、設定の名前を入力します。
5. [サンプリングレート] に、ログレコードを受信する対象のリクエストの割合を入力します。
6. [フィールド]] で、リアルタイムログで受信するフィールドを選択します。
   - すべての CMCD フィールドをログに含めるには、[CMCD のすべてのキー] を選択します。
7. [エンドポイント] で、リアルタイムログを受信する 1 つ以上の Kinesis データストリームを選択します。

> **Note**
>
> CloudFront のリアルタイムログは、Kinesis Data Streams で指定したデータストリームに配信されます。リアルタイムログを読み取って分析するには、独自の Kinesis データストリームコンシューマーを構築できます。Firehose を使用して、ログデータを Amazon S3、Amazon Redshift、Amazon OpenSearch Service、またはサードパーティーのログ処理サービスに送信することもできます。

8. [IAM ロール] で、[新しいサービスロールを作成] を選択するか、既存のロールを選択します。ただし、IAM ロールを作成するアクセス許可が必要です。
9. (オプション) [ディストリビューション] で、リアルタイムログ設定にアタッチする CloudFront ディストリビューションとキャッシュビヘイビアを選択します。
10. [Create configuration] (設定を作成) をクリックします。

正常に終了すると、作成したリアルタイムログ設定の詳細がコンソールに表示されます。

詳細については、「リアルタイムログ設定について」を参照してください。

リアルタイムログ設定を作成する (AWS CLI)

AWS Command Line Interface (AWS CLI) を使用してリアルタイムログ設定を作成するには、aws cloudfront create-realtime-log-config コマンドを使用します。コマンドの入力パラメータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

リアルタイムログ設定を作成するには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、`rtl-config.yaml` コマンドのすべての入力パラメータを含む create-realtime-log-config という名前のファイルを作成します。

   ```
   aws cloudfront create-realtime-log-config --generate-cli-skeleton yaml-input > rtl-config.yaml
   ```

2. 先ほど作成した `rtl-config.yaml` という名前のファイルを開きます。ファイルを編集して、必要なリアルタイムログ設定を指定し、ファイルを保存します。次の点に注意してください。

   - `StreamType` では、唯一の有効な値は、`Kinesis` です。

   リアルタイムロング設定の詳細については、「リアルタイムログ設定について」を参照してください。

3. 次のコマンドを使用して、`rtl-config.yaml` ファイルの入力パラメータを使用してリアルタイムログ設定を作成します。

   ```
   aws cloudfront create-realtime-log-config --cli-input-yaml file://rtl-config.yaml
   ```

成功した場合、このコマンドの出力には、先ほど作成したリアルタイムログ設定の詳細が表示されます。

リアルタイムログ設定を既存のディストリビューション (入力ファイル付き CLI) にアタッチするには

1. 以下のコマンドを使用して、更新する CloudFront ディストリビューションのディストリビューション設定を保存します。*`distribution_ID`* をディストリビューションの ID に置き換えます。

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml > dist-
config.yaml
```

2. 先ほど作成した dist-config.yaml という名前のファイルを開きます。ファイルを編集し、リアルタイムログ設定を使用するように更新する各キャッシュ動作に次の変更を加えます。

   - キャッシュ動作で、RealtimeLogConfigArn という名前のフィールドを追加します。フィールドの値には、このキャッシュ動作にアタッチするリアルタイムログ設定の ARN を使用します。
   - ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。

   完了したら、ファイルを保存します。

3. リアルタイムログ設定を使用するようにディストリビューションを更新するには、次のコマンドを使用します。*distribution_ID* をディストリビューションの ID に置き換えます。

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

成功した場合、コマンドの出力には、先ほど更新したディストリビューションの詳細が表示されます。

リアルタイムログ設定 (API) の作成

CloudFront API を使用してリアルタイムログ設定を作成するには、CreateRealtimeLogConfig を使用します。これらの API コールで指定するパラメータの詳細については、「リアルタイムログ設定について」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

リアルタイムログ設定を作成したら、次の API コールのいずれかを使用して、それをキャッシュ動作にアタッチできます。

- 既存のディストリビューションのキャッシュ動作にアタッチするには、UpdateDistribution を使用します。
- 新しいディストリビューションのキャッシュ動作にアタッチするには、CreateDistribution を使用します。

これらの API コールの両方について、キャッシュ動作内で、RealtimeLogConfigArn フィールドにリアルタイムログ設定の ARN を指定します。これらの API コールで指定するその他フィールドの詳細については、「ディストリビューション設定リファレンス」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

Kinesis Data Streams コンシューマーの作成

リアルタイムログを読み取って分析するには、Kinesis Data Streams コンシューマーを構築または使用します。CloudFront リアルタイムログ用のコンシューマーを構築する場合、すべてのリアルタイムログレコードのフィールドは、フィールド セクションの一覧と常に同じ順序で配信されることを知っておくことが重要です。この固定注文に対応するためにコンシューマーを構築することを確認してください。

たとえば、time-to-first-byte、sc-status、および c-country の 3 つのフィールドのみを含むリアルタイムログ設定を考えてみます。このシナリオでは、最後のフィールド、c-country は、すべてのログレコードで常にフィールド番号 3 です。ただし、後でリアルタイムログ設定にフィールドを追加すると、レコード内の各フィールドの配置が変更される可能性があります。

たとえば、フィールド sc-bytes と time-taken をリアルタイムログ設定に追加した場合、これらのフィールドは、フィールド セクションに示されている順序に従って各ログレコードに挿入されます。5 つのフィールドすべての順序は time-to-first-byte、sc-status、sc-bytes、time-taken、および c-country です。c-country フィールドはもともとフィールド番号 3 でしたが、現在はフィールド番号 5 です。リアルタイムログ設定にフィールドを追加する場合は、コンシューマーアプリケーションがログレコード内の位置を変更するフィールドを処理できることを確認してください。

リアルタイムログのトラブルシューティング

リアルタイムログ設定を作成した後、レコードが Kinesis Data Streams にまったく配信されない (または一部のレコードが配信されない) 場合があります。この場合、まず CloudFront ディストリビューションがビューワーリクエストを受信していることを確認する必要があります。その場合は、次の設定を確認してトラブルシューティングを続行できます。

IAM ロールのアクセス許可

CloudFront では、リアルタイムログレコードを Kinesis データストリームに配信するために、リアルタイムログ設定の IAM ロールが使用されます。ロールの信頼ポリシーとロールのアクセス許可ポリシーが、IAM ロール に示されているポリシーと一致していることを確認してください。

Kinesis Data Streams のスロットリング

CloudFront によってリアルタイムログレコードが Kinesis データストリームに書き込まれる速度が、ストリームで処理できる速度を上回る場合、Kinesis Data Streams は CloudFront からのリクエストを抑制することがあります。この場合、Kinesis データストリームのシャードの数を増やすことができます。各シャードは、1 秒あたり 1,000 レコードまでの書き込みをサポートし、1 秒あたり 1 MB の最大データ書き込みをサポートします。

エッジ関数のログ

Amazon CloudWatch Logs を使用して、Lambda@Edge 関数と CloudFront Functions の両方のエッジ関数のログを取得できます。ログにアクセスするには、CloudWatch コンソールまたは CloudWatch Logs API を使用できます。

Important

ログは、すべてのリクエストを完全に課金するためのものではなく、コンテンツに対するリクエストの本質を把握するものとして使用することをお勧めします。CloudFront はベストエフォートベースでエッジ関数のログを提供します。特定のリクエストのログエントリが、リクエストが実際に処理されてからかなり後に配信されることも、(まれに) 一切配信されないこともあります。ログエントリをエッジ関数のログから省略すると、エッジ関数のログ内のエントリ数は AWS の請求と使用状況レポートに表示される使用量と一致しなくなります。

Lambda@Edge のログ

Lambda@Edge は、関数ログを CloudWatch Logs に自動的に送信し、関数が実行される AWS リージョン にログストリームを作成します。ロググループ名は `/aws/lambda/us-east-1.`*`function-name`* として書式化されます。ここで、*`function-name`* は、作成時に関数に付けた名前であり、`us-east-1` は、関数が作成された AWS リージョン のリージョンコードです。関数が実行される他のリージョンのロググループであっても、ロググループ名には常に `us-east-1` が含まれます。

Note

Lambda@Edge は、リクエストのボリュームとログのサイズに基づいてログを調整します。

Lambda@Edge 関数ログファイルを確認するには、正しい AWS リージョン で CloudWatch ログファイルを確認する必要があります。Lambda@Edge 関数が実行されているリージョンを確認するには、CloudFront コンソールでその関数のメトリクスのグラフを表示します。メトリクスは AWS リージョン ごとに表示されます。同じページで、リージョンを選択してそのリージョンのログファイルを表示し、問題を調査することができます。

Lambda@Edge 関数で CloudWatch Logs を使用する方法の詳細については、以下を参照してください。

- CloudFront コンソールの [Monitoring (モニタリング)] セクションでのグラフ表示の詳細については、「the section called “Amazon CloudWatch による CloudFront メトリクスのモニタリング”」を参照してください。
- CloudWatch Logs にデータを送信するために必要なアクセス許可については、「the section called “IAM アクセス許可のセットアップ”」を参照してください。
- Lambda@Edge 関数のログ作成の追加については、AWS Lambda デベロッパーガイドの「Node.js の AWS Lambda 関数ログ作成」または「Python の AWS Lambda 関数ログ作成」を参照してください。
- CloudWatch Logs クォータ (以前は制限と呼ばれていました) の詳細については、Amazon CloudWatch Logs ユーザーガイドの「CloudWatch Logs クォータ」を参照してください。

CloudFront Functions のログ

CloudFront 関数のコードに `console.log()` ステートメントが含まれている場合、CloudFront Functions はこれらのログ行を CloudWatch Logs に自動的に送信します。`console.log()` ステートメントがない場合、CloudWatch Logs には何も送信されません。

CloudFront Functions は、関数が実行されたエッジロケーションに関係なく、常に米国東部 (バージニア北部) リージョン (`us-east-1`) にログストリームを作成します。ロググループ名の形式は `/aws/cloudfront/function/`*`FunctionName`*、*`FunctionName`* は関数を作成したときに指定した名前です。ログストリーム名の形式は `YYYY/M/D/`*`UUID`* です。

CloudWatch Logs に送信されるログメッセージの例を次に示します。各行は、CloudFront リクエストを一意に識別する ID で始まります。メッセージは、CloudFront ディストリビューション ID を含む START 行で始まり、END 行で終わります。START 行と END 行の間には、関数の `console.log()` ステートメントによって生成されるログ行があります。

```
U7b4hR_RaxMADupvKAvr8_m9gsGXvioUggLV5Oyq-vmAtH8HADpjhw== START DistributionID:
 E3E5D42GADAXZZ
```

```
U7b4hR_RaxMADupvKAvr8_m9gsGXvioUggLV5Oyq-vmAtH8HADpjhw== Example function log output
U7b4hR_RaxMADupvKAvr8_m9gsGXvioUggLV5Oyq-vmAtH8HADpjhw== END
```

> **Note**
>
> CloudFront Functions は、本番リクエストとレスポンスに応答して実行されるステージ LIVE の関数のみ、ログを CloudWatch に送信します。関数をテストするとき、CloudFront は CloudWatch にログを送信しません。テスト出力には、エラー、コンピューティング使用率、関数ログ (console.log() ステートメント) に関する情報が含まれていますが、この情報は CloudWatch に送信されません。

CloudFront Functions は、AWS Identity and Access Management (IAM) サービスリンクロールを使用して、アカウントの CloudWatch Logs にログを送信します。サービスにリンクされたロールは、AWS のサービスに直接リンクされた IAM ロールです。サービスにリンクされたロールは、サービスによって事前定義されており、お客様の代わりにサービスから他の AWS サービスを呼び出す必要のあるアクセス許可がすべて含まれています。CloudFront Functions は、AWSServiceRoleForCloudFrontLogger と呼ばれるサービスリンクされたロールを使用します。このロールの詳細については、「the section called “Lambda@Edge 用のサービスにリンクされたロール”」を参照してください (Lambda@Edge は同じサービスリンクされたロールを使用します)。

関数が検証エラーまたは実行エラーで失敗すると、CloudFront の標準ログとリアルタイムログに情報が記録されます。エラーに関する情報は x-edge-result-type、x-edge-response-result-type、x-edge-detailed-result-type の各フィールドに記録されます。

AWS CloudTrail を使用した Amazon CloudFront API コールのログ記録

CloudFront は、ユーザー、ロール、または AWS のサービス が実行したアクションの記録を提供するサービスである AWS CloudTrail と統合されています。CloudTrail は、CloudFront のすべての API コールをイベントとしてキャプチャします。このキャプチャには、CloudFront コンソールからの呼び出しと、CloudFront API オペレーションへのコード呼び出しが含まれます。CloudTrail で収集した情報を使用して、CloudFront に対するリクエスト、リクエスト元の IP アドレス、リクエストの作成日時、その他の詳細を確認できます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます:

- ルートユーザーまたはユーザー認証情報のどちらを使用してリクエストが送信されたか
- リクエストが IAM Identity Center ユーザーに代わって行われたかどうか。

- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービス によって送信されたかどうか。

アカウントを作成すると、AWS アカウント で CloudTrail がアクティブになり、自動的に CloudTrail の[イベント履歴] にアクセスできるようになります。CloudTrail の [イベント履歴] では、AWS リージョン で過去 90 日間に記録された 管理イベントの表示、検索、およびダウンロードが可能で、変更不可能な記録を確認できます。詳細については、「AWS CloudTrail ユーザーガイド」の「CloudTrail イベント履歴の使用」を参照してください。[イベント履歴] の閲覧には CloudTrail の料金はかかりません。

AWS アカウント で過去 90 日間のイベントを継続的に記録するには、証跡または CloudTrail Lake イベントデータストアを作成します。

CloudTrail 証跡

証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。AWS Management Console を使用して作成した証跡はマルチリージョンです。AWS CLI を使用する際は、単一リージョンまたは複数リージョンの証跡を作成できます。アカウント内のすべて AWS リージョン でアクティビティを把握するため、マルチリージョン証跡を作成することをお勧めします。単一リージョンの証跡を作成する場合、証跡の AWS リージョン に記録されたイベントのみを表示できます。証跡の詳細については、「AWS CloudTrail ユーザーガイド」の「AWS アカウント の証跡の作成」および「組織の証跡の作成」を参照してください。

証跡を作成すると、進行中の管理イベントのコピーを 1 つ無料で CloudTrail から Amazon S3 バケットに配信できますが、Amazon S3 ストレージには料金がかかります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。Amazon S3 の料金に関する詳細については、「Amazon S3 の料金」を参照してください。

CloudTrail Lake イベントデータストア

CloudTrail Lake を使用すると、イベントに対して SQL ベースのクエリを実行できます。CloudTrail Lake は、行ベースの JSON 形式の既存のイベントを Apache ORC 形式に変換します。ORC は、データを高速に取得するために最適化された単票ストレージ形式です。イベントはイベントデータストアに集約されます。イベントデータストアは、高度なイベントセレクタを適用することによって選択する条件に基いた、イベントのイミュータブルなコレクションです。どのイベントが存続し、クエリに使用できるかは、イベントデータストアに適用するセレクタが制御します。CloudTrail Lake の詳細については、「AWS CloudTrail ユーザーガイド」の「Lake の使用AWS CloudTrail」を参照してください。

CloudTrail Lake のイベントデータストアとクエリにはコストがかかります。イベントデータストアを作成する際に、イベントデータストアに使用する料金オプションを選択します。料金オプションによって、イベントの取り込みと保存にかかる料金、および、そのイベントデータストアのデフォルトと最長の保持期間が決まります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。

> **Note**
>
> CloudFront はグローバルサービスです。CloudTrail は、CloudFront のイベントを米国東部 (バージニア北部) リージョンに記録します。詳細については、AWS CloudTrail ユーザーガイドの「グローバルサービスイベント」を参照してください。
> AWS Security Token Service で一時的なセキュリティ認証情報を使用すると、`us-west-2` などのリージョンエンドポイントへの呼び出しは、CloudTrail で適切なリージョンに記録されます。
> CloudFront エンドポイントの詳細については、「AWS 全般のリファレンス」の「CloudFront のエンドポイントとクォータ」を参照してください。

CloudTrail の CloudFront データイベント

データイベントは、リソースで実行されたリソースオペレーション (CloudFront ディストリビューションに対する読み取りや書き込みなど) に関する情報を提供します。これらのイベントは、データプレーンオペレーションとも呼ばれます。データイベントは、多くの場合、高ボリュームのアクティビティです。デフォルトでは、CloudTrail はデータイベントをログ記録しません。CloudTrail [イベント履歴] にはデータイベントは記録されません。

追加の変更がイベントデータに適用されます。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。

CloudFront リソースタイプのデータイベントは、CloudTrail コンソール、AWS CLI、または CloudTrail API オペレーションを使用してログに記録できます。データイベントをログに記録する方法の詳細については、「 AWS CloudTrailユーザーガイド」の「 AWS Management Consoleを使用したデータイベントのログ記録」および「AWS Command Line Interfaceを使用したデータイベントのログ記録」を参照してください。

次の表に、データイベントをログに記録できる CloudFront リソースタイプを示します。データイベントタイプ (コンソール) 列には、CloudTrail コンソールの[データイベントタイプ]リストから選択する値が表示されます。resources.type 値列には、AWS CLI または CloudTrail API を使用して高度

なイベントセレクタを設定するときに指定する `resources.type` 値が表示されます。CloudTrail に記録されたデータ API 列には、リソース タイプの CloudTrail にログ記録された API コールが表示されます。

データイベントタイプ (コンソール)	resources.type 値	CloudTrail にログ記録されたデータ API
CloudFront KeyValueStore	`AWS::CloudFront::KeyValueStore`	• DeleteKeys • DescribeKeyValueStore • GetKey • ListKeys • PutKeys • UpdateKeys

`eventName`、`readOnly`、および `resources.ARN` フィールドでフィルタリングして、自分にとって重要なイベントのみを口グに記録するように高度なイベントセレクタを設定できます。オブジェクトの詳細については、「AWS CloudTrail API リファレンス」の「AdvancedFieldSelector」を参照してください。

CloudTrail の CloudFront 管理イベント

管理イベントでは、AWS アカウント のリソースに対して実行される管理オペレーションについての情報が得られます。これらのイベントは、コントロールプレーンオペレーションとも呼ばれます。CloudTrail は、デフォルトで管理イベントをログ記録します。

Amazon CloudFront は、すべての CloudFront コントロールプレーンオペレーションを管理イベントとして記録します。CloudFront が CloudTrail に記録する Amazon CloudFront コントロールプレーンオペレーションのリストについては、「Amazon CloudFront API リファレンス」を参照してください。

CloudFront イベントの例

各イベントは任意の送信元からの単一のリクエストを表し、リクエストされた API オペレーション、オペレーションの日時、リクエストパラメーターなどに関する情報を含みます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、イベントは特定の順序で表示されません。

目次

例：UpdateDistribution

次は、UpdateDistribution オペレーションを示す CloudTrail イベントの例です。

CloudFront API への呼び出しの場合、`eventSource` は `cloudfront.amazonaws.com` です。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:role-session-name",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/role-session-name",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-02-02T19:23:50Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2024-02-02T19:26:01Z",
    "eventSource": "cloudfront.amazonaws.com",
    "eventName": "UpdateDistribution",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "52.94.133.137",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/121.0.0.0 Safari/537.36",
    "requestParameters": {
        "distributionConfig": {
```

```
            "defaultRootObject": "",
            "aliases": {
                "quantity": 3,
                "items": [
                    "alejandro_rosalez.awsps.myinstance.com",
                    "cross-testing.alejandro_rosalez.awsps.myinstance.com",
                    "*.alejandro_rosalez.awsps.myinstance.com"
                ]
            },
            "cacheBehaviors": {
                "quantity": 0,
                "items": []
            },
            "httpVersion": "http2and3",
            "originGroups": {
                "quantity": 0,
                "items": []
            },
            "viewerCertificate": {
                "minimumProtocolVersion": "TLSv1.2_2021",
                "cloudFrontDefaultCertificate": false,
                "aCMCertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                "sSLSupportMethod": "sni-only"
            },
            "webACLId": "arn:aws:wafv2:us-east-1:111122223333:global/webacl/testing-
acl/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "customErrorResponses": {
                "quantity": 0,
                "items": []
            },
            "logging": {
                "includeCookies": false,
                "prefix": "",
                "enabled": false,
                "bucket": ""
            },
            "priceClass": "PriceClass_All",
            "restrictions": {
                "geoRestriction": {
                    "restrictionType": "none",
                    "quantity": 0,
                    "items": []
                }
```

```
            },
            "isIPV6Enabled": true,
            "callerReference": "1578329170895",
            "continuousDeploymentPolicyId": "",
            "enabled": true,
            "defaultCacheBehavior": {
                "targetOriginId": "d111111abcdef8",
                "minTTL": 0,
                "compress": false,
                "maxTTL": 31536000,
                "functionAssociations": {
                    "quantity": 0,
                    "items": []
                },
                "trustedKeyGroups": {
                    "quantity": 0,
                    "items": [],
                    "enabled": false
                },
                "smoothStreaming": false,
                "fieldLevelEncryptionId": "",
                "defaultTTL": 86400,
                "lambdaFunctionAssociations": {
                    "quantity": 0,
                    "items": []
                },
                "viewerProtocolPolicy": "redirect-to-https",
                "forwardedValues": {
                    "cookies": {"forward": "none"},
                    "queryStringCacheKeys": {
                        "quantity": 0,
                        "items": []
                    },
                    "queryString": false,
                    "headers": {
                        "quantity": 1,
                        "items": ["*"]
                    }
                },
                "trustedSigners": {
                    "items": [],
                    "enabled": false,
                    "quantity": 0
                },
```

```
                "allowedMethods": {
                    "quantity": 2,
                    "items": [
                        "HEAD",
                        "GET"
                    ],
                    "cachedMethods": {
                        "quantity": 2,
                        "items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                }
            },
            "staging": false,
            "origins": {
                "quantity": 1,
                "items": [
                    {
                        "originPath": "",
                        "connectionTimeout": 10,
                        "customOriginConfig": {
                            "originReadTimeout": 30,
                            "hTTPSPort": 443,
                            "originProtocolPolicy": "https-only",
                            "originKeepaliveTimeout": 5,
                            "hTTPPort": 80,
                            "originSslProtocols": {
                                "quantity": 3,
                                "items": [
                                    "TLSv1",
                                    "TLSv1.1",
                                    "TLSv1.2"
                                ]
                            }
                        },
                        "id": "d111111abcdef8",
                        "domainName": "d111111abcdef8.cloudfront.net",
                        "connectionAttempts": 3,
                        "customHeaders": {
                            "quantity": 0,
                            "items": []
                        },
```

```
                        "originShield": {"enabled": false},
                        "originAccessControlId": ""
                    }
                ]
            },
            "comment": "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "id": "EDFDVBD6EXAMPLE",
        "ifMatch": "E1RTLUR9YES76O"
    },
    "responseElements": {
        "distribution": {
            "activeTrustedSigners": {
                "quantity": 0,
                "enabled": false
            },
            "id": "EDFDVBD6EXAMPLE",
            "domainName": "d111111abcdef8.cloudfront.net",
            "distributionConfig": {
                "defaultRootObject": "",
                "aliases": {
                    "quantity": 3,
                    "items": [
                        "alejandro_rosalez.awsps.myinstance.com",
                        "cross-testing.alejandro_rosalez.awsps.myinstance.com",
                        "*.alejandro_rosalez.awsps.myinstance.com"
                    ]
                },
                "cacheBehaviors": {"quantity": 0},
                "httpVersion": "http2and3",
                "originGroups": {"quantity": 0},
                "viewerCertificate": {
                    "minimumProtocolVersion": "TLSv1.2_2021",
                    "cloudFrontDefaultCertificate": false,
                    "aCMCertificateArn": "arn:aws:acm:us-
east-1:111122223333:certificate/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                    "sSLSupportMethod": "sni-only",
                    "certificateSource": "acm",
                    "certificate": "arn:aws:acm:us-east-1:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
                },
                "webACLId": "arn:aws:wafv2:us-east-1:111122223333:global/webacl/
testing-acl/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
                "customErrorResponses": {"quantity": 0},
```

```
                "logging": {
                    "includeCookies": false,
                    "prefix": "",
                    "enabled": false,
                    "bucket": ""
                },
                "priceClass": "PriceClass_All",
                "restrictions": {
                    "geoRestriction": {
                        "restrictionType": "none",
                        "quantity": 0
                    }
                },
                "isIPV6Enabled": true,
                "callerReference": "1578329170895",
                "continuousDeploymentPolicyId": "",
                "enabled": true,
                "defaultCacheBehavior": {
                    "targetOriginId": "d111111abcdef8",
                    "minTTL": 0,
                    "compress": false,
                    "maxTTL": 31536000,
                    "functionAssociations": {"quantity": 0},
                    "trustedKeyGroups": {
                        "quantity": 0,
                        "enabled": false
                    },
                    "smoothStreaming": false,
                    "fieldLevelEncryptionId": "",
                    "defaultTTL": 86400,
                    "lambdaFunctionAssociations": {"quantity": 0},
                    "viewerProtocolPolicy": "redirect-to-https",
                    "forwardedValues": {
                        "cookies": {"forward": "none"},
                        "queryStringCacheKeys": {"quantity": 0},
                        "queryString": false,
                        "headers": {
                            "quantity": 1,
                            "items": ["*"]
                        }
                    },
                    "trustedSigners": {
                        "enabled": false,
                        "quantity": 0
```

```
                    },
                    "allowedMethods": {
                        "quantity": 2,
                        "items": [
                            "HEAD",
                            "GET"
                        ],
                        "cachedMethods": {
                            "quantity": 2,
                            "items": [
                                "HEAD",
                                "GET"
                            ]
                        }
                    }
                },
                "staging": false,
                "origins": {
                    "quantity": 1,
                    "items": [
                        {
                            "originPath": "",
                            "connectionTimeout": 10,
                            "customOriginConfig": {
                                "originReadTimeout": 30,
                                "hTTPSPort": 443,
                                "originProtocolPolicy": "https-only",
                                "originKeepaliveTimeout": 5,
                                "hTTPPort": 80,
                                "originSslProtocols": {
                                    "quantity": 3,
                                    "items": [
                                        "TLSv1",
                                        "TLSv1.1",
                                        "TLSv1.2"
                                    ]
                                }
                            },
                            "id": "d111111abcdef8",
                            "domainName": "d111111abcdef8.cloudfront.net",
                            "connectionAttempts": 3,
                            "customHeaders": {"quantity": 0},
                            "originShield": {"enabled": false},
                            "originAccessControlId": ""
```

```
                        }
                    ]
                },
                "comment": "HIDDEN_DUE_TO_SECURITY_REASONS"
            },
            "aliasICPRecordals": [
                {
                    "cNAME": "alejandro_rosalez.awsps.myinstance.com",
                    "iCPRecordalStatus": "APPROVED"
                },
                {
                    "cNAME": "cross-testing.alejandro_rosalez.awsps.myinstance.com",
                    "iCPRecordalStatus": "APPROVED"
                },
                {
                    "cNAME": "*.alejandro_rosalez.awsps.myinstance.com",
                    "iCPRecordalStatus": "APPROVED"
                }
            ],
            "aRN": "arn:aws:cloudfront::111122223333:distribution/EDFDVBD6EXAMPLE",
            "status": "InProgress",
            "lastModifiedTime": "Feb 2, 2024 7:26:01 PM",
            "activeTrustedKeyGroups": {
                "enabled": false,
                "quantity": 0
            },
            "inProgressInvalidationBatches": 0
        },
        "eTag": "E1YHBLAB2BJY1G"
    },
    "requestID": "4e6b66f9-d548-11e3-a8a9-73e33example",
    "eventID": "5ab02562-0fc5-43d0-b7b6-90293example",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "apiVersion": "2020_05_31",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "cloudfront.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
```

```
}
```

例：UpdateKeys

次は、UpdateKeys オペレーションを示す CloudTrail イベントの例です。

CloudFront KeyValueStore API への呼び出しの場合、eventSource は cloudfront.amazonaws.com ではなく、edgekeyvaluestore.amazonaws.com になります。

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:role-session-name",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/role-session-name",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2023-11-01T23:41:14Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-11-01T23:41:28Z",
    "eventSource": "edgekeyvaluestore.amazonaws.com",
    "eventName": "UpdateKeys",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "3.235.183.252",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/121.0.0.0 Safari/537.36,
    "requestParameters": {
        "kvsARN": "arn:aws:cloudfront::111122223333:key-value-store/a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111",
        "ifMatch": "KV3O6B1CX531EBP",
        "deletes": [
```

```
            {"key": "key1"}
        ]
    },
    "responseElements": {
        "itemCount": 0,
        "totalSizeInBytes": 0,
        "eTag": "KVDC9VEVZ71ZGO"
    },
    "requestID": "5ccf104c-acce-4ea1-b7fc-73e33example",
    "eventID": "a0b1b5c7-906c-439d-9925-90293example",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::CloudFront::KeyValueStore",
            "ARN": "arn:aws:cloudfront::111122223333:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "111122223333.cloudfront-kvs.global.api.aws"
    }
}
```


CloudTrail レコードの内容については、「AWS CloudTrail ユーザーガイド」の「CloudTrail record contents」を参照してください。

AWS Config による設定変更の追跡

AWS Config を使用して、CloudFront ディストリビューションの設定変更を記録できます。ディストリビューションの状態、料金クラス、オリジン、地域制限の設定、および Lambda@Edge 設定の変更をキャプチャできます。

> **Note**
>
> AWS Config では、CloudFront ストリーミングディストリビューションのキーと値のタグは記録されません。

CloudFront で AWS Config をセットアップする

AWS Config を設定するとき、サポートされるすべての AWS リソースを記録するか、または特定のリソースのみを指定して設定の変更を記録するか (CloudFront の変更のみを記録する、など) を選択できます。サポートされる CloudFront リソースのリストについては、「AWS Config デベロッパーガイド」の「サポートされるリソースタイプ」トピックの「Amazon CloudFront」セクションを参照してください。

CloudFront ディストリビューションの設定変更を追跡するには、米国東部 (バージニア北部) AWS リージョン で CloudFront コンソールにサインする必要があります。

> **Note**
>
> AWS Config でのリソースの記録には遅延が生じる可能性があります。AWS Config は、リソースを検出した後でしかリソースを記録しません。

Console

CloudFront を使用して AWS Config を設定するには

1. AWS Management Console にサインインして、AWS Config コンソール (https://console.aws.amazon.com/config/) を開きます。
2. [Get Started Now] を選択します。
3. [設定] ページの [記録するリソースタイプ] で、AWS で記録する AWS Config リソースタイプを指定します。CloudFront の変化のみを記録する場合には、[特定の型] を選択し、[CloudFront] で、変更を追跡するディストリビューションまたはストリーミング配信を選択します。

   追跡するディストリビューションを追加あるいは変更するには、最初のステップを完了した後に左側で [設定] を選択します。

4. AWS Config で追加の必須オプションを指定する: 通知の設定、設定情報の場所の指定、リソースタイプ評価のルールの追加。

詳細については、AWS Config デベロッパーガイドの「コンソールによる AWS Config の設定」を参照してください。

AWS CLI

AWS CLI を使用して CloudFront で AWS Config をセットアップするには、「AWS Config デベロッパーガイド」の「AWS CLI を使用した AWS Config のセットアップ」を参照してください。

AWS Config API

AWS Config API を使用して CloudFront で AWS Config をセットアップするには、「AWS Config API リファレンス」の StartConfigurationRecorder アクションおよびその他の情報を参照してください。

CloudFront 設定履歴の表示

AWS Config がディストリビューションへの設定変更の記録を開始したら、CloudFront 用に設定した任意のディストリビューションの設定履歴を取得できます。

設定履歴は以下のいずれかの方法で表示できます。

Console

記録されたリソースごとに、設定の詳細の履歴を示すタイムラインページを表示できます。このページを表示するには、[Dedicated Hosts] ページの [設定タイムライン] 列にあるグレーのアイコンを選択します。

詳細については、『AWS Config デベロッパーガイド』の「AWS Config コンソールでの設定詳細の表示」を参照してください。

AWS CLI

すべてのディストリビューションのリストを取得するには、次の例に示されているように、list-discovered-resources コマンドを使用します。

```
aws configservice list-discovered-resources --resource-type
 AWS::CloudFront::Distribution
```

特定の期間のディストリビューションの設定詳細を取得するには、get-resource-config-history コマンドを使用します。

詳細については、『AWS Config デベロッパーガイド』の「CLI による設定詳細の表示」を参照してください。

AWS Config API

すべてのディストリビューションのリストを取得するには、ListDiscoveredResources アクションを使用します。

特定の期間のディストリビューションの設定詳細を取得するには、GetResourceConfigHistory アクションを使用します。詳細については、「AWS Config API リファレンス」を参照してください。

Amazon CloudFront のセキュリティ

AWS ではクラウドセキュリティが最優先事項です。セキュリティを最も重視する組織の要件を満たすために構築された AWS のデータセンターとネットワークアーキテクチャは、お客様に大きく貢献します。

セキュリティは、AWS と顧客の間の責任共有です。責任共有モデルでは、この責任がクラウドのセキュリティおよびクラウド内のセキュリティとして説明されています。

- クラウドのセキュリティ - AWS は、AWS クラウドで AWS のサービスを実行するインフラストラクチャを保護する責任を負います。また、AWS は、使用するサービスを安全に提供します。AWS コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon CloudFront に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラムによる AWS 対象範囲内のサービス」を参照してください。
- クラウド内のセキュリティ - お客様の責任は、使用する AWS のサービスに応じて異なります。お客様は、データの機密性、組織の要件、および適用法令と規制などのその他要因に対する責任も担います。

このドキュメントは、CloudFront を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するように CloudFront を設定する方法を示します。また、CloudFront リソースのモニタリングや保護に AWS の他のサービスを利用する方法についても説明します。

トピック

Amazon CloudFront におけるデータ保護

AWS の責任共有モデルは、Amazon CloudFront でのデータ保護に適用されます。このモデルで説明されているように、AWS は、AWS クラウド のすべてを実行するグローバルインフラストラクチャを保護するがあります。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービスのセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「データプライバシーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された記事「AWS 責任共有モデルおよび GDPR」を参照してください。

データを保護するため、AWS アカウント 認証情報を保護し、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーをセットアップすることをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須であり TLS 1.3 がお勧めです。
- AWS CloudTrail で API とユーザーアクティビティロギングをセットアップします。
- AWS のサービス 内のすべてのデフォルトセキュリティ管理に加え、AWS 暗号化ソリューションを使用します。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API により AWS にアクセスするときに FIPS 140-2 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

お客様のメールアドレスなどの機密情報や重要な情報を、タグや自由形式のテキストフィールド (名前フィールドなど) に入力しないよう強くお勧めします。これは、コンソール、API、AWS CLI、または AWS SDK を使用して CloudFront やその他の AWS のサービスを使用する場合も同様です。名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

Amazon CloudFront には、配信するコンテンツを保護するために使用できるいくつかのオプションがあります。

- HTTPS 接続を設定します。

- フィールドレベルの暗号化を構成して、転送中の特定のデータのセキュリティを強化します。
- コンテンツへのアクセスを制限して、特定の人々、または特定のエリアのユーザーだけが閲覧できるようにします。

以下のトピックでオプションについて詳しく説明します。

トピック

転送時の暗号化

転送中にデータを暗号化するには、Amazon CloudFront がビューワーと通信するときに接続が暗号化されるように、ビューワーが HTTPS を使用してファイルをリクエストするように CloudFront を設定します。また、オリジンからファイルを取得する際に CloudFront が HTTPS を使用するように設定すると、CloudFront とオリジンとの通信で接続が暗号化されます。

詳細については、「CloudFront で HTTPS を使用する」を参照してください。

フィールドレベル暗号化では、HTTPS と共にセキュリティのレイヤーが追加されます。これにより、システムの処理中に特定のデータに特定のアプリケーションのみがアクセスできるように、そのデータを保護できます。CloudFront でフィールドレベルの暗号化を設定することで、ユーザーが送信した機密情報をウェブサーバーに安全にアップロードできます。クライアントが提供した機密情報は、ユーザーに近いエッジで暗号化され、アプリケーションスタック全体で暗号化されたままになります。これにより、データを必要とするアプリケーションのみが (復号化するための認証情報があれば) そのデータを復号化できます。

詳細については、「フィールドレベル暗号化を使用した機密データの保護」を参照してください。

CloudFront API エンドポイント (`cloudfront.amazonaws.com`、`cloudfront-fips.amazonaws.com`) は HTTPS トラフィックのみ受け入れます。つまり、CloudFront API を使用して情報を送受信すると、データ (ディストリビューション設定、キャッシュポリシーとオリジンリクエストポリシー、キーグループとパブリックキー、CloudFront Functions の関数コードなど) は常に転送中に暗号化されます。さらに、CloudFront API エンドポイントに送信されたすべてのリクエストは、AWS 認証情報で署名され、AWS CloudTrail に記録されます。

CloudFront Functions の関数コードと設定は、エッジロケーション POP (Points Of Presence)、および CloudFront で使用される他のストレージロケーション間でコピーされると、転送時に常に暗号化されます。

保管中の暗号化

CloudFront Functions の関数コードと設定は、エッジロケーション POP や、CloudFront で使用されるその他のストレージロケーションに、常に暗号化された形式で保存されます。

コンテンツへのアクセス制限

インターネット経由でコンテンツを配信する多くの企業が、ユーザーのサブセットのドキュメント、ビジネスデータ、メディアストリーム、またはコンテンツに対して、アクセスを制限する必要があると考えています。Amazon CloudFront を使用してこのコンテンツを安全に供給するには、以下の 1 つ以上の方法を使用できます。

署名付き URL または署名付き Cookie を使用する

署名付き URL または署名付き Cookie を使用して CloudFront を介してこの非公開コンテンツを配信することで、選択したユーザー、たとえば有料のユーザーを対象としたコンテンツへのアクセスを制限できます。詳細については、「署名付き URL と署名付き Cookie を使用したプライベートコンテンツを提供する」を参照してください。

Amazon S3 バケットのコンテンツへのアクセスを制限する

例えば、CloudFront 署名付き URL または署名付き Cookie を使用してコンテンツへのアクセスを制限する場合は、ファイルの直接 URL を使用して、ユーザーがファイルを表示できないようにします。代わりに、CloudFront URL を使用してファイルへのアクセスのみ許可するため、正常に保護されます。

CloudFront ディストリビューションのオリジンとして Amazon S3 バケットを使用する場合は、S3 バケットへのアクセスを制限することができるオリジンアクセスコントロール (OAC) を設定できます。詳細については、「the section called “Amazon Simple Storage Service オリジンへのアクセスを制限する”」を参照してください。

Application Load Balancer が提供するコンテンツへのアクセスを制限する

オリジンとして、Elastic Load Balancing の Application Load Balancer で CloudFront を使用する場合、ユーザーが Application Load Balancer に直接アクセスできないように CloudFront を設定できます。これにより、ユーザーは CloudFront 経由でしか Application Load Balancer に

アクセスできず、CloudFront を使用するメリットを得ることができます。詳細については、「Application Load Balancer へのアクセスを制限する」を参照してください。

AWS WAF ウェブ ACL の使用

ウェブアプリケーションファイアウォールサービスである AWS WAF を使用して、コンテンツへのアクセスを制限するためのウェブアクセス制御リスト (ウェブ ACL) を作成できます。指定した条件 (リクエスト送信元の IP アドレス、クエリ文字列の値など) に基づいて、CloudFront はリクエストされたコンテンツを返すか、HTTP 403 ステータスコード (Forbidden) を返すことで、リクエストに応答します。詳細については、「AWS WAF 保護を使用する」を参照してください。

地域制限を使用する

地域制限 (地理的ブロッキング) を使用すると、CloudFront ディストリビューションを通じて配信しているコンテンツについて、特定地域のユーザーによるアクセスを回避できます。地域制限を設定するときに選択できるオプションがいくつかあります。詳細については、「コンテンツの地理的配分を制限する」を参照してください。

Amazon CloudFront のアイデンティティとアクセス管理

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御するために役立つ AWS のサービスです。IAM 管理者は、誰を認証 (サインイン) し、誰に CloudFront リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加費用なしで使用できる AWS のサービスです。

トピック

- 対象者
- アイデンティティによる認証
- ポリシーを使用したアクセス権の管理
- Amazon CloudFront と IAM との連携方法
- Amazon CloudFront のアイデンティティベースのポリシー例
- Amazon CloudFront の AWS 管理ポリシー
- Amazon CloudFront のアイデンティティとアクセスのトラブルシューティング

対象者

AWS Identity and Access Management (IAM) の使い方は、CloudFront で行う作業によって異なります。

サービスユーザー – ジョブを実行するために CloudFront サービスを使用する場合は、管理者から必要な認証情報とアクセス許可が与えられます。作業を実行するためにさらに多くの CloudTrail 機能を使用する場合、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解すると、管理者から適切な権限をリクエストするのに役に立ちます。CloudFront の機能にアクセスできない場合は、「Amazon CloudFront のアイデンティティとアクセスのトラブルシューティング」を参照してください。

サービス管理者 - 社内の CloudFront リソースを担当している管理者には、通常、CloudFront にフルアクセスできます。サービスユーザーが CloudFront のどの機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。CloudFront で IAM をどのように自社で利用できるかの詳細については、「Amazon CloudFront と IAM との連携方法」を参照してください。

IAM 管理者 - IAM 管理者は、CloudFront へのアクセスを管理するポリシーの詳しい作成方法を確認したい場合があります。IAM で使用できる CloudFront アイデンティティベースのポリシー例については、「Amazon CloudFront のアイデンティティベースのポリシー例」を参照してください。

アイデンティティによる認証

認証とは、アイデンティティ認証情報を使用して AWSにサインインする方法です。ユーザーは、AWS アカウントのルートユーザー として、IAM ユーザーとして、または IAM ロールを引き受けることによって、認証済み (AWS にサインイン済み) である必要があります。

ID ソースから提供された認証情報を使用すると、フェデレーティッドアイデンティティとして AWS にサインインできます。AWS IAM Identity Centerフェデレーティッドアイデンティティの例としては、(IAM アイデンティティセンター) ユーザー、貴社のシングルサインオン認証、Google または Facebook の認証情報などがあります。フェデレーティッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して AWSにアクセスする場合、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、AWS Management Console または AWSアクセスポータルにサインインできます。AWS へのサインインの詳細については、『AWS サインイン ユーザーガイド』の「AWS アカウント にサインインする方法」を参照してください。

プログラムで AWS にアクセスする場合、AWS は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) を提供し、認証情報でリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに署名する推奨方法の使用については、『IAM ユーザーガイド』の「AWS API リクエストの署名」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、AWS では、アカウントのセキュリティ強化のために多要素認証 (MFA) の使用をお勧めしています。詳細については、『AWS IAM Identity Center ユーザーガイド』の「Multi-factor authentication」(多要素認証) および『IAM ユーザーガイド』の「AWS における多要素認証 (MFA) の使用」を参照してください。

AWS アカウントのルートユーザー

AWS アカウントを作成する場合は、このアカウントのすべての AWS のサービスとリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用したメールアドレスとパスワードでサインインすることによってアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、『IAM ユーザーガイド』の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに対し、ID プロバイダーとのフェデレーションを使用して、一時認証情報の使用により、AWS のサービス にアクセスすることを要求します。

フェデレーテッドアイデンティティ は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリのユーザーか、または ID ソースから提供された認証情報を使用して AWS のサービス にアクセスするユーザーです。フェデレーテッドアイデンティティ が AWS アカウント にアクセスすると、ロールが継承され、ロールは一時認証情報を提供します。

アクセスを一元管理する場合は、AWS IAM Identity Center を使用することをお勧めします。IAM アイデンティティセンター でユーザーとグループを作成するか、すべての AWS アカウント とアプリケーションで使用するために、独自の ID ソースで一連のユーザーとグループに接続して同期することもできます。IAM Identity Center の詳細については、『AWS IAM Identity Center ユーザーガイド』の「What is IAM Identity Center?」(IAM Identity Center とは) を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、1 人のユーザーまたは 1 つのアプリケーションに対して特定の権限を持つ AWS アカウント 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、IAM ユーザーガイドの「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する権限を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、『IAM ユーザーガイド』の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定の権限を持つ、AWS アカウント 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロールの切り替えによって、AWS Management Console で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、AWS CLI または AWSAPI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッドアイデンティティに権限を割り当てるには、ロールを作成してそのロールの権限を定義します。フェデレーティッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限が付与されます。フェデレーションの詳細については、『IAM ユーザーガイド』の「サードパーティーアイデンティティプロバイダー向けロールの作成」 を参照してください。IAM アイデンティティセンターを使用する場合、権限セットを設定します。アイデンティティが認証後にアク

セスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。権限セットの詳細については、『AWS IAM Identity Center ユーザーガイド』の「権限セット」を参照してください。

- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。
- クロスサービスアクセス権 - 一部の AWS のサービスでは、他の AWS のサービスの機能を使用します。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの権限、サービスロール、またはサービスにリンクされたロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービス またはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの「AWS のサービス にアクセス許可を委任するロールの作成」を参照してください。
  - サービスリンクロール - サービスリンクロールは、AWS のサービス にリンクされたサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスリンクロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

- Amazon EC2 で実行されるアプリケーション - EC2 インスタンスで実行され、AWS CLI または AWS API 要求を行っているアプリケーションの一時的な認証情報を管理するために、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付されたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、『IAM ユーザーガイド』の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して権限を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、『IAM ユーザーガイド』の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセス権の管理

AWS でアクセス権を管理するには、ポリシーを作成して AWS アイデンティティまたはリソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けて、これらの権限を定義します。AWS は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うと、これらのポリシーを評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWSに保存されます。JSON ポリシードキュメントの構造と内容の詳細については、『IAM ユーザーガイド』の「JSON ポリシー概要」を参照してください。

管理者は AWSJSON ポリシーを使用して、だれが何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS Management Console、AWS CLI、または AWS API からロール情報を取得できます。

アイデンティティベースポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティ

ベースのポリシーを作成する方法については、IAM ユーザーガイドの「IAM ポリシーの作成」を参照してください。

アイデンティティベースポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれます。管理ポリシーは、AWS アカウント 内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンポリシーです。マネージドポリシーには、AWS マネージドポリシーおよびカスタマーマネージドポリシーがあります。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、『IAM ユーザーガイド』の「マネージドポリシーとインラインポリシーの比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービス を含めることができます。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは IAM の AWSマネージドポリシーは使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、AWS WAF、および Amazon VPC は、ACL をサポートするサービスの例です。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS では、他の一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界界の詳細については、『IAM ユーザーガイド』の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCP) - SCP は、AWS Organizations で組織や組織単位 (OU) の最大権限を指定する JSON ポリシーです。AWS Organizations は、顧客のビジネスが所有する複数の AWS アカウント をグループ化し、一元的に管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP はメンバーアカウントのエンティティに対する権限を制限します (各 AWS アカウントのルートユーザー など)。Organizations と SCP の詳細については、『AWS Organizations ユーザーガイド』の「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、「IAM ユーザーガイド」の「ポリシーの評価論理」を参照してください。

Amazon CloudFront と IAM との連携方法

IAM を使用して CloudFront へのアクセスを管理する前に、CloudFront で利用できる IAM の機能を確認してください。

Amazon CloudFront で利用できる IAM の機能

IAM の機能	CloudFront サポート
アイデンティティベースのポリシー	Yes
リソースベースのポリシー	No
ポリシーアクション	Yes
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	No
ABAC (ポリシー内のタグ)	部分的
一時的な認証情報	はい
転送アクセスセッション (FAS)	いいえ
サービスロール	いいえ
サービスリンクロール	はい

大部分の IAM 機能が CloudFront やその他の AWS のサービスとどのように連携するかの概要については、IAM ユーザーガイドの「IAM と連携する AWS のサービス」を参照してください。

CloudFront のアイデンティティベースのポリシー

アイデンティティベースポリシーをサポートする	Yes

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素については、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

CloudFront のアイデンティティベースのポリシー例

CloudFront のアイデンティティベースのポリシー例については、「Amazon CloudFront のアイデンティティベースのポリシー例」を参照してください。

CloudFront 内のリソースベースのポリシー

リソースベースのポリシーのサポート	No

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービス を含めることができます。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる AWS アカウント にある場合、信頼できるアカウントの IAM 管理者は、リソースにアクセスするための権限をプリンシパルエンティティ (ユーザーまたはロール) に付与する必要もあります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーを追加する必要はありません。詳細については、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

CloudFront のポリシーアクション

ポリシーアクションに対するサポート	Yes

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーの`Action`要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連する AWSAPI オペレーションと同じです。一致する API オペレーションのない権限のみのアクションなど、いくつかの例外があります。また、ポリシーに複数アクションが必要なオペレーションもあります。これらの追加アクションは、 依存アクション と呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

CloudFront アクションのリストを確認するには、サービス認証リファレンスの「Amazon CloudFront で定義されるアクション」を参照してください。

CloudFront のポリシーアクションは、アクションの前に次のプレフィックスを使用します。

```
cloudfront
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "cloudfront:action1",
      "cloudfront:action2"
         ]
```

CloudFront のアイデンティティベースのポリシーの例については、「Amazon CloudFront のアイデンティティベースのポリシー例」を参照してください。

CloudFront のポリシーリソース

ポリシーリソースに対するサポート	Yes

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースにどのような条件でアクションを実行できるかということです。

Resource JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの権限と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

CloudFront リソースのタイプとその ARN のリストを確認するには、サービス認証リファレンスの「Amazon CloudFront で定義されるリソース」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Amazon CloudFront で定義されるアクション」を参照してください。

CloudFront のアイデンティティベースのポリシーの例については、「Amazon CloudFront のアイデンティティベースのポリシー例」を参照してください。

CloudFront のポリシー条件キー

サービス固有のポリシー条件キーのサポート	はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成することで、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の Condition 要素を指定するか、1 つの Condition 要素に複数のキーを指定すると、AWS は AND 論理演算子を使用してそれらを評価します。単一の条件キーに複数

の値を指定すると、AWS は OR 論理演算子を使用して条件を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、『IAM ユーザーガイド』の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS はグローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、IAM ユーザーガイド の「AWS グローバル条件コンテキストキー」を参照してください。

CloudFront の条件キーのリストを確認するには、サービス認証リファレンスの「Amazon CloudFront の条件キー」を参照してください。どのアクションやリソースで条件キーを使用できるかについては、「Amazon CloudFront で定義されるアクション」を参照してください。

CloudFront のアイデンティティベースのポリシー例については、「Amazon CloudFront のアイデンティティベースのポリシー例」を参照してください。

CloudFront の ACL

ACL のサポート	No

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

CloudFront での ABAC

ABAC (ポリシー内のタグ) のサポート	部分的

属性ベースのアクセスコントロール (ABAC) は、属性に基づいて権限を定義する認可戦略です。AWS では、属性は タグ と呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール)、および多数の AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合に操作を許可するように ABAC ポリシーを設計します。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、aws:ResourceTag/*key-name*、aws:RequestTag/*key-name*、または aws:TagKeys の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値ははいです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、『IAM ユーザーガイド』の「ABAC とは?」を参照してください。ABAC をセットアップするステップを説明するチュートリアルについては、「IAM ユーザーガイド」の「属性に基づくアクセスコントロール (ABAC) を使用する」を参照してください。

CloudFront は、 ディストリビューションの ABAC のみをサポートします。

CloudFront での一時的な認証情報の使用

一時的な認証情報のサポート	Yes

AWS のサービスには、一時認証情報を使用してサインインしても機能しないものがあります。一時的な認証情報を利用できる AWS のサービス を含めた詳細情報については、『IAM ユーザーガイド』の「IAM と連携する AWS のサービス」を参照してください。

ユーザー名とパスワード以外の方法で AWS Management Consoleにサインインする場合は、一時認証情報を使用していることになります。例えば、会社のシングルサインオン (SSO) リンクを使用して AWS にアクセスすると、そのプロセスは自動的に一時認証情報を作成します。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ロールへの切り替え (コンソール)」を参照してください。

一時認証情報は、AWS CLI または AWSAPI を使用して手動で作成できます。作成後、一時認証情報を使用して AWSにアクセスできるようになります。AWS は、長期的なアクセスキーを使用する代わりに、一時認証情報を動的に生成することをお勧めします。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

CloudFront の転送アクセスセッション

転送アクセスセッション (FAS) をサポート	いいえ

IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行してから、別のサービスの別のアクションを開始することがあります。FAS は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービス またはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

CloudFront のサービスロール

サービスロールのサポート	いいえ

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

> **Warning**
>
> サービスロールのアクセス許可を変更すると、CloudFront の機能が破損する可能性があります。CloudFront が指示する場合以外は、サービスロールを編集しないでください。

CloudFront のサービスにリンクされたロール

サービスリンクロールのサポート	Yes

サービスリンクロールは、AWS のサービス にリンクされているサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスリンクロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

Lambda@Edge は、サービスにリンクされたロールを使用してアクションを実行します。CloudFront サービスにリンクされたロールの作成または管理の詳細については、「Lambda@Edge 用のサービスにリンクされたロール」を参照してください。

サービスにリンクされたロールの作成または管理の詳細については、「IAM と提携する AWS のサービス」を参照してください。表の中から、[Service-linked role] (サービスにリンクされたロール) 列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] リンクを選択します。

Amazon CloudFront のアイデンティティベースのポリシー例

デフォルトでは、ユーザーおよびロールには、CloudFront リソースを作成または変更するアクセス許可がありません。また、AWS Management Console、AWS Command Line Interface (AWS CLI)、または AWS API を使用してタスクを実行することもできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

CloudFront が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、サービス認証リファレンスの「Amazon CloudFront のアクション、リソース、および条件キー」を参照してください。

トピック

- ポリシーのベストプラクティス
- CloudFront コンソールを使用
- 自分の権限の表示をユーザーに許可する
- プログラムで CloudFront にアクセスするための許可
- CloudFront コンソールの使用に必要なアクセス許可
- CloudFront 向けの AWS マネージド (事前定義) ポリシー
- カスタマーマネージドポリシーの例

ポリシーのベストプラクティス

アイデンティティベースのポリシーは、アカウント内で、CloudFront リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションを実行すると、AWS アカウント に料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS マネージドポリシーを使用して開始し、最小特権の権限に移行する – ユーザーとワークロードへの権限の付与を開始するには、多くの一般的なユースケースのために権限を付与する AWS マネージドポリシーを使用します。これらは AWS アカウントで使用できます。ユースケースに応じた AWS カスタマーマネージドポリシーを定義することで、権限をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイドの「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで権限を設定するときは、タスクの実行に必要な権限のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権権限とも呼ばれています。IAM を使用して権限を適用する方法の詳細については、『IAM ユーザーガイド』の「IAM でのポリシーと権限」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。また、AWS CloudFormation などの特定の AWS のサービスを介して使用する場合、条件を使ってサービスアクションへのアクセス権を付与することもできます。詳細については、IAM ユーザーガイドの [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素：条件) を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer ポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する – AWS アカウント で IAM ユーザーまたはルートユーザーを要求するシナリオがある場合は、セキュリティを強化するために MFA をオンにします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA 保護 API アクセスの設定」を参照してください。

IAM でのベストプラクティスの詳細については、『IAM ユーザーガイド』の「IAM でのセキュリティのベストプラクティス」を参照してください。

CloudFront コンソールを使用

Amazon CloudFront コンソールにアクセスするには、最小限のアクセス許可が必要です。これらのアクセス許可により、AWS アカウントの CloudFront リソースの詳細を一覧表示できます。最小限必要なアクセス許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポ

リシーを持つエンティティ (ユーザーまたはロール) ではコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソール権限を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスを許可します。

ユーザーとロールが引き続き CloudFront コンソールを使用できるようにするには、エンティティに *ConsoleAccess* または *ReadOnly* AWS 管理ポリシーもアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI か AWS API を使用してプログラム的に、このアクションを完了する権限が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
```

```
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

プログラムで CloudFront にアクセスするための許可

以下に示しているのは、アクセス権限ポリシーです。Sid (ステートメント ID) はオプションです。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowAllCloudFrontPermissions",
         "Effect": "Allow",
         "Action": ["cloudfront:*"],
         "Resource": "*"
      }
   ]
}
```

ポリシーでは、すべての CloudFront オペレーションを実行するアクセス許可を付与します。これは、プログラムで CloudFront にアクセスするのに十分なアクセス許可です。コンソールを使用して CloudFront にアクセスする場合は、「CloudFront コンソールの使用に必要なアクセス許可」を参照してください。

各アクションを使用するアクセス許可を付与または拒否するために指定するアクションおよび ARN のリストについては、サービス認証リファレンスの「Amazon CloudFront のアクション、リソース、および条件キー」を参照してください。

CloudFront コンソールの使用に必要なアクセス許可

CloudFront コンソールへのフルアクセスを許可するには、以下のアクセス許可ポリシーでアクセス許可を付与します。

```
{
```

```
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm:ListCertificates",
            "cloudfront:*",
            "cloudwatch:DescribeAlarms",
            "cloudwatch:PutMetricAlarm",
            "cloudwatch:GetMetricStatistics",
            "elasticloadbalancing:DescribeLoadBalancers",
            "iam:ListServerCertificates",
            "sns:ListSubscriptionsByTopic",
            "sns:ListTopics",
            "waf:GetWebACL",
            "waf:ListWebACLs"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets",
            "s3:PutBucketPolicy"
         ],
         "Resource":"arn:aws:s3:::*"
      }
   ]
}
```

アクセス許可が必要な理由は次のとおりです。

`acm:ListCertificates`

CloudFront コンソールを使用してディストリビューションを作成および更新していて、ビューワーと CloudFront 間、または CloudFront とオリジン間で HTTPS を必須とするように CloudFront を設定するときに、ACM 証明書のリストを表示できるようにします。

CloudFront コンソールを使用していない場合、このアクセス許可は必要ありません。

`cloudfront:*`

すべての CloudFront アクションを実行できるようにします。

`cloudwatch:DescribeAlarms` と `cloudwatch:PutMetricAlarm`

CloudFront コンソールで CloudWatch アラームを作成および表示できるようにします。`sns:ListSubscriptionsByTopic` と `sns:ListTopics` も参照してください。

CloudFront コンソールを使用していない場合、これらのアクセス許可は必要ありません。

`cloudwatch:GetMetricStatistics`

CloudFront コンソールで CloudWatch メトリクスを表示できるようにします。

CloudFront コンソールを使用していない場合、このアクセス許可は必要ありません。

`elasticloadbalancing:DescribeLoadBalancers`

ディストリビューションを作成および更新するときに、使用できるオリジンのリストで、Elastic Load Balancing ロードバランサーのリストを表示できるようにします。

CloudFront コンソールを使用していない場合、このアクセス許可は必要ありません。

`iam:ListServerCertificates`

CloudFront コンソールを使用してディストリビューションを作成および更新していて、ビューワーと CloudFront 間、または CloudFront とオリジン間で HTTPS を必須とするように CloudFront を設定するときに、IAM 証明書ストアで証明書のリストを表示できるようにします。

CloudFront コンソールを使用していない場合、このアクセス許可は必要ありません。

`s3:ListAllMyBuckets`

ディストリビューションを作成および更新するときに、以下のオペレーションを実行できるようにします。

- 使用できるオリジンのリストで S3 バケットのリストを表示する
- アクセスログを保存できる S3 バケットのリストを表示する

CloudFront コンソールを使用していない場合、このアクセス許可は必要ありません。

`S3:PutBucketPolicy`

S3 バケットへのアクセスを制限するディストリビューションを作成または更新するときに、CloudFront オリジンのアクセスアイデンティティにアクセスできるようにバケットポリシーを更新することをユーザーに許可します。詳しくは、「the section called “オリジンアクセスアイデンティティを使用する (レガシー、非推奨)”」を参照してください。

CloudFront コンソールを使用していない場合、このアクセス許可は必要ありません。

`sns:ListSubscriptionsByTopic` と `sns:ListTopics`

CloudFront コンソールで CloudWatch アラームを作成するときに、通知用の SNS トピックを選択できるようにします。

CloudFront コンソールを使用していない場合、これらのアクセス許可は必要ありません。

`waf:GetWebACL` と `waf:ListWebACLs`

CloudFront コンソールで AWS WAF ウェブ ACL のリストを表示できるようにします。

CloudFront コンソールを使用していない場合、これらのアクセス許可は必要ありません。

CloudFront 向けの AWS マネージド (事前定義) ポリシー

AWS は、 によって作成され管理されるスタンドアロンの IAM ポリシーを提供することで、多くの一般的ユースケースに対応します。AWSこれらの AWS 管理ポリシーは、一般的ユースケースに必要なアクセス権限を付与することで、どの権限が必要なのかをユーザーが調査する必要をなくすことができます。詳細については、IAM ユーザーガイドの「AWS 管理ポリシー」を参照してください。CloudFront 用に、IAM には 2 つの管理ポリシーが用意されています。

- CloudFrontFullAccess - CloudFront リソースへのフルアクセスを許可します。

> **Important**
>
> CloudFront でアクセスログを作成および保存するには、追加のアクセス許可を付与する必要があります。詳細については、「標準ログ記録の設定およびログファイルへのアクセスに必要なアクセス許可」を参照してください。

- CloudFrontReadOnlyAccess - CloudFront リソースへの読み取り専用アクセスを許可します。

カスタマーマネージドポリシーの例

独自のカスタム IAM ポリシーを作成して、CloudFront API アクションに対するアクセス許可を付与できます。これらのカスタムポリシーは、指定されたアクセス許可が必要な IAM ユーザーまたはグループにアタッチできます。これらのポリシーは、CloudFront API、AWS SDK、または AWS CLI を使用しているときに機能します。次の例では、いくつかの一般的ユースケースのアクセス権限を示します。CloudFront へのフルアクセスをユーザーに許可するポリシーについては、「CloudFront コンソールの使用に必要なアクセス許可」を参照してください。

例

- 例 1: すべてのディストリビューションへの読み取りアクセスを許可する
- 例 2: ディストリビューションの作成、更新、削除を許可する
- 例 3: 無効化の作成と一覧表示を許可する
- 例 4: ディストリビューションの作成を許可する

例 1: すべてのディストリビューションへの読み取りアクセスを許可する

以下のアクセス許可ポリシーでは、CloudFront コンソールですべてのディストリビューションを表示するアクセス許可をユーザーに付与します。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm:ListCertificates",
            "cloudfront:GetDistribution",
            "cloudfront:GetDistributionConfig",
            "cloudfront:ListDistributions",
            "cloudfront:ListCloudFrontOriginAccessIdentities",
            "elasticloadbalancing:DescribeLoadBalancers",
            "iam:ListServerCertificates",
            "sns:ListSubscriptionsByTopic",
            "sns:ListTopics",
            "waf:GetWebACL",
            "waf:ListWebACLs"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets"
         ],
         "Resource":"arn:aws:s3:::*"
      }
   ]
}
```

例 2: ディストリビューションの作成、更新、削除を許可する

以下のアクセス許可ポリシーでは、CloudFront コンソールを使用して、ディストリビューションを作成、更新、削除することをユーザーに許可します。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm:ListCertificates",
            "cloudfront:CreateDistribution",
            "cloudfront:DeleteDistribution",
            "cloudfront:GetDistribution",
            "cloudfront:GetDistributionConfig",
            "cloudfront:ListDistributions",
            "cloudfront:UpdateDistribution",
            "cloudfront:ListCloudFrontOriginAccessIdentities",
            "elasticloadbalancing:DescribeLoadBalancers",
            "iam:ListServerCertificates",
            "sns:ListSubscriptionsByTopic",
            "sns:ListTopics",
            "waf:GetWebACL",
            "waf:ListWebACLs"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets",
            "s3:PutBucketPolicy"
         ],
         "Resource":"arn:aws:s3:::*"
      }
   ]
}
```

`cloudfront:ListCloudFrontOriginAccessIdentities` アクセス許可では、ユーザーは、Amazon S3 バケットのオブジェクトを操作するアクセス許可が既存のオリジンアクセスアイデンティティに自動的に付与されるようにできます。また、ユーザーがオリジンアクセスアイデンティティを作成できるようにする場合

は、cloudfront:CreateCloudFrontOriginAccessIdentity アクセス権限も許可する必要があります。

例 3: 無効化の作成と一覧表示を許可する

次のアクセス権限ポリシーでは、ユーザーが無効化を作成し、一覧表示できるようにします。これに CloudFront ディストリビューションへの読み取りアクセスが含まれているのは、最初にディストリビューション用の設定を表示して、無効化を作成および表示するためです。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm:ListCertificates",
            "cloudfront:GetDistribution",
            "cloudfront:GetStreamingDistribution",
            "cloudfront:GetDistributionConfig",
            "cloudfront:ListDistributions",
            "cloudfront:ListCloudFrontOriginAccessIdentities",
            "cloudfront:CreateInvalidation",
            "cloudfront:GetInvalidation",
            "cloudfront:ListInvalidations",
            "elasticloadbalancing:DescribeLoadBalancers",
            "iam:ListServerCertificates",
            "sns:ListSubscriptionsByTopic",
            "sns:ListTopics",
            "waf:GetWebACL",
            "waf:ListWebACLs"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets"
         ],
         "Resource":"arn:aws:s3:::*"
      }
   ]
}
```

例 4: ディストリビューションの作成を許可する

次のアクセス許可ポリシーでは、CloudFront コンソールでディストリビューションを作成して表示するためのアクセス許可をユーザーに付与します。CreateDistribution アクションでは、ディストリビューション ARN (arn:aws:cloudfront::123456789012:distribution/*) のワイルドカードの代わりに、Resource のワイルドカード (*) 文字を指定します。Resource 要素の詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素: Resource」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "cloudfront:CreateDistribution",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "cloudfront:ListDistributions",
            "Resource": "*"
        }
    ]
}
```

Amazon CloudFront の AWS 管理ポリシー

ユーザー、グループ、ロールにアクセス許可を追加するには、自分でポリシーを作成するよりも、AWS 管理ポリシーを使用する方が簡単です。ユーザーに必要なアクセス許可のみを提供する IAM カスタマー管理ポリシーを作成するには、時間と専門知識が必要です。すぐに使用を開始するために、AWS マネージドポリシーを使用できます。これらのポリシーは、一般的なユースケースをターゲット範囲に含めており、AWS アカウント で利用できます。AWS マネージドポリシーの詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS のサービスは、AWS マネージドポリシーを維持および更新します。AWS 管理ポリシーのアクセス許可を変更することはできません。サービスでは、新しい機能を利用できるようにするために、AWS マネージドポリシーに権限が追加されることがあります。この種類の更新は、ポリシーが

アタッチされている、すべてのアイデンティティ (ユーザー、グループおよびロール) に影響を与えます。新しい機能が立ち上げられたり、新しいアクセス許可が使用可能になったりすると、各サービスで AWS 管理ポリシーが更新される可能性が最も高くなります。各サービスでは、AWS 管理ポリシーからアクセス許可が削除されないため、ポリシーの更新によって既存のアクセス許可が破棄されることはありません。

さらに、AWS は、複数のサービスにまたがるジョブ機能の特徴に対するマネージドポリシーもサポートしています。例えば、ReadOnlyAccess AWS マネージドポリシーでは、すべての AWS のサービスおよびリソースへの読み取り専用アクセスを許可します。サービスが新しい機能を起動する場合、AWS は、新たなオペレーションとリソース用に、読み取り専用の許可を追加します。ジョブ関数ポリシーのリストと説明については、IAM ユーザーガイドの「AWS ジョブ関数のマネージドポリシー」を参照してください。

AWS 管理ポリシー: CloudFrontReadOnlyAccess

IAM ID に CloudFrontReadOnlyAccess ポリシーをアタッチできます。このポリシーでは CloudFront リソースへの読み取り専用アクセス許可が許可されます。また、CloudFront に関連し、CloudFront コンソールに表示される他の AWS サービスのリソースへの読み取り専用アクセスも許可されます。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `cloudfront:Describe*` - プリンシパルが CloudFront リソースに関するメタデータに関する情報を取得できるようにします。
- `cloudfront:Get*` - プリンシパルが CloudFront リソースの詳細情報と設定を取得できるようにします。
- `cloudfront:List*` - プリンシパルが CloudFront リソースのリストを取得できるようにします。
- `cloudfront-keyvaluestore:Describe*` - プリンシパルに、キー値ストアに関する情報の取得を許可します。
- `cloudfront-keyvaluestore:Get*` - プリンシパルに、キー値ストアの詳細情報や設定の取得を許可します。
- `cloudfront-keyvaluestore:List*` - プリンシパルに、キー値ストアのリストの取得を許可します。
- `acm:ListCertificates` - プリンシパルが ACM 証明書のリストを取得できるようにします。

- iam:ListServerCertificates - プリンシパルが IAM に格納されるサーバー証明書のリストを許可できるようにします。
- route53:List* - プリンシパルが Route 53 リソースのリストを取得できるようにします。
- waf:ListWebACLs - プリンシパルが AWS WAF のウェブ ACL のリストを取得できるようにします。
- waf:GetWebACL - プリンシパルが AWS WAF のウェブ ACL に関する詳細情報を取得できるようにします。
- wafv2:ListWebACLs - プリンシパルが AWS WAF のウェブ ACL のリストを取得できるようにします。
- wafv2:GetWebACL - プリンシパルが AWS WAF のウェブ ACL に関する詳細情報を取得できるようにします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "cfReadOnly",
      "Effect": "Allow",
      "Action": [
        "acm:ListCertificates",
        "cloudfront:Describe*",
        "cloudfront:Get*",
        "cloudfront:List*",
        "cloudfront-keyvaluestore:Describe*",
        "cloudfront-keyvaluestore:Get*",
        "cloudfront-keyvaluestore:List*",
        "iam:ListServerCertificates",
        "route53:List*",
        "waf:ListWebACLs",
        "waf:GetWebACL",
        "wafv2:ListWebACLs",
        "wafv2:GetWebACL"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS 管理ポリシー: CloudFrontFullAccess

IAM ID に CloudFrontFullAccess ポリシーをアタッチできます。このポリシーでは、CloudFront リソースに対する管理アクセス許可が許可されます。また、CloudFront に関連し、CloudFront コンソールに表示される他の AWS サービスのリソースへの読み取り専用アクセスも許可されます。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `s3:ListAllMyBuckets` - プリンシパルが、すべての Amazon S3 バケットのリストを取得できるようにします。
- `acm:ListCertificates` - プリンシパルが ACM 証明書のリストを取得できるようにします。
- `cloudfront:*` - プリンシパルが、すべての CloudFront リソースに対してすべてのアクションを実行できるようにします。
- `cloudfront-keyvaluestore:*` - プリンシパルに、キー値ストアに対するすべてのアクションの実行を許可します。
- `iam:ListServerCertificates` - プリンシパルが IAM に格納されるサーバー証明書のリストを許可できるようにします。
- `waf:ListWebACLs` - プリンシパルが AWS WAF のウェブ ACL のリストを取得できるようにします。
- `waf:GetWebACL` - プリンシパルが AWS WAF のウェブ ACL に関する詳細情報を取得できるようにします。
- `wafv2:ListWebACLs` - プリンシパルが AWS WAF のウェブ ACL のリストを取得できるようにします。
- `wafv2:GetWebACL` - プリンシパルが AWS WAF のウェブ ACL に関する詳細情報を取得できるようにします。
- `kinesis:ListStreams` - プリンシパルが Amazon Kinesis ストリームのリストを取得できるようにします。
- `kinesis:DescribeStream` - プリンシパルが Kinesis ストリームに関する詳細情報を取得できるようにします。
- `iam:ListRoles` - プリンシパルが IAM のロールのリストを取得できるようにします。

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

```
  "Statement": [
    {
      "Sid": "cfflistbuckets",
      "Action": [
        "s3:ListAllMyBuckets"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "cffullaccess",
      "Action": [
        "acm:ListCertificates",
        "cloudfront:*",
        "cloudfront-keyvaluestore:*",
        "iam:ListServerCertificates",
        "waf:ListWebACLs",
        "waf:GetWebACL",
        "wafv2:ListWebACLs",
        "wafv2:GetWebACL",
        "kinesis:ListStreams"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "cffdescribestream",
      "Action": [
        "kinesis:DescribeStream"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:kinesis:*:*:*"
    },
    {
      "Sid": "cfflistroles",
      "Action": [
        "iam:ListRoles"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:*"
    }
  ]
}
```

AWS 管理ポリシー: AWSCloudFrontLogger

IAM ID に AWSCloudFrontLogger ポリシーをアタッチすることはできません。このポリシーは、CloudFront がユーザーに代わってアクションを実行できるようにする、サービスにリンクされたロールにアタッチされます。詳細については、「the section called “Lambda@Edge 用のサービスにリンクされたロール”」を参照してください。

このポリシーにより、CloudFront が Amazon CloudWatch にログファイルをプッシュできるようになります。このポリシーに含まれるアクセス許可の詳細については、「the section called “CloudFront ロガー用のサービスにリンクされたロールのアクセス許可”」を参照してください。

AWS 管理ポリシー: AWSLambdaReplicator

IAM ID に AWSLambdaReplicator ポリシーをアタッチすることはできません。このポリシーは、CloudFront がユーザーに代わってアクションを実行できるようにする、サービスにリンクされたロールにアタッチされます。詳細については、「the section called “Lambda@Edge 用のサービスにリンクされたロール”」を参照してください。

このポリシーにより、CloudFront が AWS Lambda の関数を作成、削除、無効化して AWS リージョン に Lambda@Edge 関数をレプリケートできるようになります。このポリシーに含まれるアクセス許可の詳細については、「the section called “Lambda Replicator 用のサービスにリンクされたロールのアクセス許可”」を参照してください。

CloudFront による AWS 管理ポリシーの更新

CloudFront の AWS 管理ポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。このページへの変更に関する自動通知については、CloudFront ドキュメントの履歴ページの RSS フィードを購読してください。

変更	説明	日付
CloudFrontReadOnlyAccess と CloudFrontFullAccess - 2 つの既存のポリシーに対する更新	CloudFront はキー値ストアに対する新しいアクセス許可を追加しました。 新しいアクセス許可により、ユーザーはキー値ストアに関する情報を取得し、キー値ス	2023 年 12 月 19 日

変更	説明	日付
	トアにアクションを実行できます。	
CloudFrontReadOnlyAccess – 既存ポリシーへの更新	CloudFront に CloudFront Functions を記述するためのアクセス許可が新たに追加されました。 このアクセス許可により、ユーザー、グループ、またはロールは関数に関する情報とメタデータを読み取ることができます。ただし、関数のコードを読み取ることはできません。	2021 年 9 月 8 日
CloudFront が変更の追跡を開始しました	CloudFront が AWS 管理ポリシーの変更の追跡を開始しました。	2021 年 9 月 8 日

Amazon CloudFront のアイデンティティとアクセスのトラブルシューティング

以下の情報は、CloudFront と IAM の使用時に発生する可能性がある一般的な問題の診断や修復に役立ちます。

トピック

- CloudFront でアクションを実行する権限がありません
- iam:PassRole を実行する権限がありません
- 自分の AWS アカウント外のユーザーに CloudFront リソースへのアクセスを許可したい

CloudFront でアクションを実行する権限がありません

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要なcloudfront:*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 cloudfront:GetWidget on resource: my-example-widget
```

この場合、cloudfront:*GetWidget* アクションを使用して *my-example-widget*リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。管理者とは、サインイン認証情報を提供した担当者です。

iam:PassRole を実行する権限がありません

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して CloudFront にロールを渡すことができるようにする必要があります。

一部の AWS のサービスでは、新しいサービスロールまたはサービスリンクロールを作成せずに、既存のロールをサービスに渡すことが許可されています。そのためには、サービスにロールを渡す権限が必要です。

以下の例に示すエラーは、marymajor という名前の IAM ユーザーがコンソールを使用して CloudFront でアクションを実行しようとした場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。Mary には、ロールをサービスに渡す権限がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。管理者とは、サインイン認証情報を提供した担当者です。

自分の AWS アカウント外のユーザーに CloudFront リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまた

はアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください。

- CloudFront でこれらの機能がサポートされているかどうかを確認するには、「Amazon CloudFront と IAM との連携方法」を参照してください。
- 所有している AWS アカウント全体のリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「所有している別の AWS アカウントへのアクセス権を IAM ユーザーに提供」を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、『IAM ユーザーガイド』の「第三者が所有する AWS アカウント へのアクセス権を付与する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、『IAM ユーザーガイド』の「外部で認証されたユーザー (ID フェデレーション) へのアクセス権限」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

Amazon CloudFront でのログ記録とモニタリング

モニタリングは、CloudFront および AWS ソリューションの可用性とパフォーマンスを維持する上で重要な役割を果たします。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできるように、AWS ソリューションのすべての部分からのモニタリングデータを収集する必要があります。AWS は、CloudFront リソースとアクティビティをモニタリングし、潜在的なインシデントに対応するための複数のツールを提供します。

Amazon CloudWatch アラーム

Amazon CloudWatch アラームを使用して、指定した期間にわたって 1 つのメトリクスを確認します。メトリクスが特定のしきい値を超えると、Amazon SNS のトピックまたは AWS Auto Scaling ポリシーに通知が送信されます。メトリクスが特定の状態にある場合、CloudWatch アラームはアクションを呼び出しません。状態が変わり、それが指定した期間だけ維持される必要があります。詳細については、「Amazon CloudWatch による CloudFront メトリクスのモニタリング」を参照してください。

AWS CloudTrail ログ

CloudTrail は、CloudFront のユーザー、ロール、または AWS のサービスによって実行された API アクションの記録を提供します。CloudTrail で収集された情報を使用して、CloudFront に対する API リクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。詳しくは、「AWS CloudTrail を使用した Amazon CloudFront API コールのログ記録」を参照してください。

CloudFront 標準ログとリアルタイムログ

CloudFront は、ディストリビューションに対して行われたリクエストに関する詳細なレコードを提供します。これらのログは、多くのアプリケーションで役立ちます。たとえば、ログ情報は、セキュリティやアクセスの監査に役立ちます。詳しくは、「CloudFront とエッジ関数のログ記録」を参照してください。

エッジ関数のログ

CloudFront Functions と Lambda@Edge の両方のエッジ関数によって生成されたログは、Amazon CloudWatch Logs に直接送信され、CloudFront によってどこかに保存されることはありません。CloudFront Functions は、AWS Identity and Access Management (IAM) サービスリンクロールを使用して、カスタマー生成ログをお客様のアカウントの CloudWatch Logs に直接送信します。

CloudFront コンソールレポート

CloudFront コンソールには、キャッシュ統計レポート、人気オブジェクトレポート、およびトップ参照元レポートなど、さまざまなレポートがあります。ほとんどの CloudFront コンソールレポートは、CloudFront アクセスログのデータに基づいています。このログには、CloudFront が受信するすべてのユーザーリクエストの詳細が含まれています。ただし、このレポートを表示するために、アクセスログを有効にする必要はありません。詳細については、「コンソールで CloudFront レポートを表示する」を参照してください。

Amazon CloudFront のコンプライアンス検証

サードパーティーの監査者は、さまざまな AWS コンプライアンスプログラムの一環として Amazon CloudFront のセキュリティとコンプライアンスを評価します。このプログラムには、SOC、PCI、HIPAA などが含まれます。

特定のコンプライアンスプログラムの対象範囲に含まれる AWS のサービスのリストについては、「コンプライアンスプログラムによる対象範囲内の AWS のサービス」を参照してください。一般的な情報については、「AWS コンプライアンスプログラム」を参照してください。

AWS Artifact を使用して、サードパーティーの監査レポートをダウンロードできます。詳細については、「AWS Artifact にレポートをダウンロードする」を参照してください。

CloudFront を使用する際のコンプライアンス責任は、データの機密性、企業のコンプライアンス目標、適用法規や規則によって決まります。AWS ではコンプライアンスに役立つ以下のリソースを用意しています。

- セキュリティおよびコンプライアンスのクイックスタートガイド - これらのデプロイメントガイドでは、アーキテクチャー上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境を AWS にデプロイするための手順を説明します。
- AWS での HIPAA のセキュリティとコンプライアンスの設計 - このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明しています。

  AWS HIPAA コンプライアンスプログラムには、HIPAA 対応サービスとして CloudFront (CloudFront Embedded POP によるコンテンツ配信を除く) が含まれています。AWS と事業提携契約 (BAA) を締結している場合は、CloudFront (CloudFront Embedded POP によるコンテンツ配信を除く) を使用して保護医療情報 (PHI) を含むコンテンツを迅速に提供できます。詳細については、「HIPAA コンプライアンス」を参照してください。
- AWS コンプライアンスのリソース - このワークブックとガイドのコレクションは、お客様の業界や所在地に適用される場合があります。
- AWS Config - この AWS のサービスでは、自社プラクティス、業界ガイドライン、および規制に対するリソースの設定の準拠状態を評価します。
- AWS Security Hub – この AWS のサービスは、セキュリティコントロールを使用してリソース設定とセキュリティ標準を評価し、お客様がさまざまなコンプライアンスフレームワークに準拠できるようサポートします。Security Hub を使用して CloudFront リソースを評価する方法の詳細については、「AWS Security Hub ユーザーガイド」の「Amazon CloudFront コントロール」を参照してください。

CloudFront コンプライアンスのベストプラクティス

このセクションでは、Amazon CloudFront を使用してコンテンツを配信するときのコンプライアンスに関するベストプラクティスと推奨事項について説明します。

AWS 責任共有モデルに基づいて PCI 準拠または HIPAA 準拠ワークロードを実行する場合は、将来の監査目的のために過去 365 日間の CloudFront 使用状況データのログを記録することをお勧めします。使用状況データを記録するには、以下の操作を実行できます。

- CloudFront アクセスログを有効にする 詳細については、「標準ログ (アクセスログ) の設定および使用」を参照してください。
- CloudFront API に送信されるリクエストを取得します。詳細については、「AWS CloudTrail を使用した Amazon CloudFront API コールのログ記録」を参照してください。

また、CloudFront がどのように PCI DSS、および SOC の基準に準拠しているかの詳細については、以下を参照してください。

Payment Card Industry Data Security Standard (PCI DSS)

CloudFront (CloudFront Embedded POP によるコンテンツ配信を除く) は、マーチャントまたはサービスプロバイダーによるクレジットカードデータの処理、ストレージ、伝送をサポートしており、ペイメントカード業界データセキュリティ基準 (PCI DSS) への準拠が検証済みです。PCI DSS の詳細 (AWS PCI Compliance Package のコピーをリクエストする方法など) については、「PCI DSS レベル 1」を参照してください。

セキュリティを確保するためのベストプラクティスとして、CloudFront エッジキャッシュにクレジットカード情報を入れないことをお勧めします。たとえば、クレジットカード番号の末尾 4 桁の数字やカード所有者の連絡先情報などのクレジットカード情報が含まれている `Cache-Control:no-cache="field-name"` ヘッダーをレスポンスに含めるようにオリジンを設定できます。

System and Organization Controls (SOC)

CloudFront (CloudFront Embedded POP によるコンテンツ配信を除く) は、SOC 1、SOC 2、SOC 3 などのシステムおよび組織管理 (SOC) 対策に準拠しています。SOC レポートは、重要なコンプライアンス管理および目標を AWS がどのように達成したかを実証する、独立したサードパーティーによる審査報告書です。これらの監査によって、お客様のデータや企業データのセキュリティ、機密保持、アベイラビリティーに影響を及ぼす可能性のあるリスクから守るために、適切な安全策と手順を講じます。これらサードパーティー監査の結果は、AWS SOC コンプライアンスのウェブサイトで参照できます。このサイトでは、AWS の業務とコンプライアンスをサポートするために制定された統制についてさらに詳しく知ることのできる発行済みレポートを公開しています。

Amazon CloudFront の耐障害性

AWS のグローバルインフラストラクチャは、AWS リージョンとアベイラビリティーゾーンを中心として構築されています。AWSリージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立し隔離されたアベイラビリティーゾーン

があります。アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、およびスケーラビリティが優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

CloudFront のオリジンフェイルオーバー

Amazon CloudFront は、AWS グローバルインフラストラクチャのサポートに加えて、データのレジリエンスニーズのサポートに役立つオリジンフェイルオーバー機能を提供しています。CloudFront は、エッジロケーションまたは POP (Point of Presence) と呼ばれるデータセンターの世界的なネットワークを通じてコンテンツを配信するグローバルサービスです。コンテンツがまだエッジロケーションにキャッシュされていない場合、CloudFront はそのコンテンツの最終版のソースとして識別したオリジンからコンテンツを取得します。

オリジンフェイルオーバーを使用して CloudFront を設定することで、回復力を向上させ、特定のシナリオの可用性を向上させることができます。開始するには、CloudFront のプライマリオリジンと 2 番目のオリジンを指定するオリジングループを作成します。プライマリオリジンが特定の HTTP ステータスコードの失敗の応答を返すと、CloudFront は自動的に 2 番目のオリジンに切り替えます。詳細については、「CloudFront オリジンフェイルオーバーを使用して高可用性を最適化する」を参照してください。

Amazon CloudFront のインフラストラクチャセキュリティ

マネージドサービスである Amazon CloudFront は、AWS グローバルネットワークセキュリティによって保護されています。AWS セキュリティサービスと AWS がインフラストラクチャを保護する方法については、「AWS クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 - AWS Well-Architected Framework」の「インフラストラクチャ保護」を参照してください。

AWS が公開した API コールを使用して、ネットワーク経由で CloudFront にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2 は必須で TLS 1.3 がお勧めです。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

CloudFront Functions は、AWS アカウント間の非常に安全な隔離バリアを使用して、Spectre や Meltdown などのサイドチャネル攻撃に対してお客様の環境を保護します。Functions は、他の顧客に属するデータにアクセスしたり、データを変更したりすることはできません。関数は、ハイパースレッディングなし、専用CPUの専用シングルスレッドプロセスで実行されます。CloudFront エッジロケーション POP (Point Of Presence) では、CloudFront Functions は一度に 1 人の顧客にのみサービスを提供し、すべての顧客固有のデータは関数の実行の間にクリアされます。

トラブルシューティング

コンテンツを配信するために Amazon CloudFront をセットアップするとき、または Lambda@Edge を使用するときに発生する可能性のある一般的な問題をトラブルシューティングし、可能な解決策を見つけます。

トピック

トラブルシューティング: ディストリビューション

証明書エラーやアクセス拒否のほか、ウェブサイトやアプリケーションを使用した Amazon CloudFront ディストリビューションのセットアップ時の一般的な問題の解決方法については、以下の情報を参照してください。

トピック

CloudFront が Access Denied エラーを返します

CloudFront ディストリビューションのオリジンとして Amazon S3 バケットを使用している場合は、次の例にアクセス拒否 (403) エラーメッセージが表示されることがあります。

目次

Amazon S3 オリジンから欠落しているオブジェクトを指定しました

バケットにリクエストされたオブジェクトが存在することを確認します。オブジェクト名では大文字と小文字が区別されます。無効なオブジェクト名を入力すると、アクセス拒否エラーコードが返される可能性があります。

例えば、CloudFront チュートリアルに従って基本的なディストリビューションを作成する場合は、オリジンとして Amazon S3 バケットを作成し、サンプル `index.html` ファイルをアップロードします。

ウェブブラウザで、`https://d111111abcdef8.cloudfront.net/`*`index`*`.html` の代わりに `https://d111111abcdef8.cloudfront.net/`*`INDEX`*`.HTML` を入力すると、URL パス内の `index.html` ファイルが大文字と小文字を区別するため、同様のメッセージが表示されることがあります。

```
<Error>
<Code>AccessDenied</Code>
<Message>Access Denied</Message>
<RequestId>22Q367AHT7Y1ABCD</RequestId>
<HostId>
ABCDE/Vg+7PSNa/d/IfFQ8Fb92TGQ0KH0ZwG5iEKbc6+e06DdMS1ZW+ryB9GFRIVtS66rSSy6So=
</HostId>
</Error>
```

Amazon S3 オリジンに IAM アクセス許可がありません

オリジンドメインおよび名前として正しい Amazon S3 バケットが選択されていることを確認します。オリジン (Amazon S3) には正しいアクセス許可が必要です。

正しいアクセス許可を指定しない場合、ビューワーに対して次のアクセス拒否メッセージが表示されることがあります。

```
<Code>AccessDenied</Code>
<Message>User: arn:aws:sts::856369053181:assumed-role/OriginAccessControlRole/
EdgeCredentialsProxy+EdgeHostAuthenticationClient is not authorized to perform:
 kms:Decrypt on the resource associated with this ciphertext because the resource does
 not exist in this Region, no resource-based policies allow access, or a resource-based
 policy explicitly denies access</Message>
<RequestId>22Q367AHT7Y1ABCD/RequestId>
<HostId>
ABCDE/Vg+7PSNa/d/IfFQ8Fb92TGQ0KH0ZwG5iEKbc6+e06DdMS1ZW+ryB9GFRIVtS66rSSy6So=
```

```
</HostId>
</Error>
```

> **Note**
>
> このエラーメッセージでは、アカウント ID 856369053181 は AWS マネージドアカウントです。

Amazon S3 からコンテンツを配信し、AWS Key Management Service (AWS KMS) サービス側の暗号化 (SSE-KMS) も使用している場合は、KMS キーと Amazon S3 バケットに指定する必要がある追加の IAM アクセス許可があります。CloudFront ディストリビューションがオリジン Amazon S3 バケットの暗号化に使用される KMS キーを使用するには、これらのアクセス許可が必要です。

Amazon S3 バケットポリシーの設定により、CloudFront ディストリビューションはコンテンツ配信のために暗号化されたオブジェクトを取得できます。

Amazon S3 バケットと KMS キーのアクセス許可を確認するには

1. 使用している KMS キーが、Amazon S3 バケットがデフォルトの暗号化に使用するキーと同じであることを確認します。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「AWS KMS (SSE-KMS) によるサーバー側の暗号化の指定」を参照してください。
2. バケット内のオブジェクトが同じ KMS キーで暗号化されていることを確認します。Amazon S3 バケットから任意のオブジェクトを選択し、サーバー側の暗号化設定を確認して KMS キー ARN を検証できます。
3. Amazon S3 バケットポリシーを編集して、Amazon S3 バケットから `GetObject` API オペレーションを呼び出すアクセス許可を CloudFront に付与します。オリジンアクセスコントロールを使用する Amazon S3 バケットポリシーの例については、「S3 バケットへのアクセス許可をオリジンアクセスコントロールに付与する」を参照してください。
4. KMS キーポリシーを編集して、`Encrypt`、`Decrypt`、および `GenerateDataKey*` に対してアクションを実行するアクセス許可を CloudFront に付与します。最小特権のアクセス許可に合わせて、指定した CloudFront ディストリビューションのみがリストされたアクションを実行できるように `Condition` 要素を指定します。既存の AWS KMS ポリシーに対してポリシーをカスタマイズできます。KMS キーポリシーの例については、「SSE-KMS」を参照してください。

OAC の代わりにオリジンアクセスアイデンティティ (OAI) を使用している場合、AWS のサービスではなくアイデンティティにアクセス許可を付与するため、Amazon S3 バケットへのアクセス許可

が若干異なります。詳細については、「Amazon S3 バケット内のファイルを読み取るアクセス許可をオリジンアクセスアイデンティティに付与する」を参照してください。

それでもディストリビューション内のファイルを表示できない場合は、「ディストリビューション内のファイルを表示できません」を参照してください。

無効な認証情報を使用しているか、十分なアクセス許可がありません

アクセス拒否エラーメッセージは、AWS SCT 認証情報 (アクセスキーとシークレットキー) が正しくないか、有効期限が切れている場合、または IAM ロールまたはユーザーが CloudFront リソースでアクションを実行するために必要なアクセス許可を保有していない場合に表示されることがあります。アクセス拒否エラーメッセージの詳細については、「IAM ユーザーガイド」の「アクセス拒否エラーメッセージのトラブルシューティング」を参照してください。

IAM と CloudFront が連携する仕組みの詳細については、「Amazon CloudFront のアイデンティティとアクセス管理」を参照してください。

代替ドメイン名を追加しようとすると、CloudFront から InvalidViewerCertificate エラーが返される

代替ドメイン名 (CNAME) をディストリビューションに追加しようとするときに CloudFront が `InvalidViewerCertificate` エラーを返す場合、この問題のトラブルシューティングについては、以下の情報を参照してください。このエラーは、代替ドメイン名を正常に追加できる前に次のいずれかの問題を解決する必要があることを示している可能性があります。

以下のエラーは、代替ドメイン名を追加することを認可されているかどうかを CloudFront が確認する順序で示されています。これによって、CloudFront が返すエラーに基づいてどの確認が正常に完了したかがわかるため、問題の解決に役立ちます。

ディストリビューションにアタッチされた証明書はありません。

> 代替ドメイン名 (CNAME) を追加するには、信頼された、有効な証明書をディストリビューションにアタッチする必要があります。前提条件を確認し、これを満たす有効な証明書を取得してディストリビューションにアタッチしてから、操作をやり直してください。詳細については、「代替ドメイン名を使用するための要件」を参照してください。

アタッチした証明書の証明書チェーンに多すぎる証明書があります。

> 証明書チェーンには最大 5 つの証明書のみを含むことができます。チェーンの証明書の数を減らしてから、操作をやり直してください。

証明書チェーンには 1 つまたは複数の現在の日付には有効ではない証明書が含まれています。

追加した証明書の証明書チェーンに、証明書がまだ有効ではないあるいは有効期限切れの 1 つ以上の証明書が含まれています。証明書チェーンの証明書の [Not Valid Before] (有効期限開始日) フィールドおよび [Not Valid After] (失効日) フィールドをチェックし、リストした日付に基づいてすべての証明書が有効であることを確認します。

アタッチした証明書は、信頼される認証機関 (CA) によって署名されていません。

代替ドメイン名を確認するために CloudFront にアタッチした証明書を自己署名証明書にすることはできません。信頼される認証機関によって署名される必要があります。詳細については、「代替ドメイン名を使用するための要件」を参照してください。

アタッチした証明書が、正しくフォーマットされない

証明書に含まれているドメイン名と IP アドレス形式、および証明書自体の形式は、証明書の標準に従っている必要があります。

CloudFront の内部エラーが発生しました。

CloudFront が内部の問題によってブロックされ、証明書を検証できませんでした。このシナリオでは、CloudFront は HTTP 500 ステータスコードを返し、証明書のアタッチについて CloudFront 内部の問題があることを示します。数分間待機してから、証明書に代替ドメイン名の追加を再試行します。

アタッチした証明書が、追加しようとしている代替ドメイン名を対象としていません。

追加する代替ドメイン名ごとに、そのドメイン名を使用することを認可されているかどうかを検証するために、CloudFront は、ドメイン名を対象とする信頼された認証機関 (CA) の有効な SSL/TLS 証明書のアタッチを求めます。証明書を更新して、追加しようとしている CNAME を対象とするドメイン名を含めてください。ワイルドカードがあるドメイン名を使用する詳細と例については、「代替ドメイン名を使用するための要件」を参照してください。

ディストリビューション内のファイルを表示できません

CloudFront ディストリビューション内のファイルを表示できない場合の一般的な解決方法を以下のトピックで説明します。

CloudFront と Amazon S3 の両方にサインアップしましたか?

Amazon S3 オリジンで Amazon CloudFront を使用するには、CloudFront と Amazon S3 の両方に個別にサインアップする必要があります。CloudFront と Amazon S3 のサインアップの詳細について、「セットアップする」を参照してください。

Amazon S3 バケットとオブジェクトのアクセス許可は正しく設定されていますか?

CloudFront を Amazon S3 オリジンで使用する場合、コンテンツのオリジナルバージョンは S3 バケットに保存されます。Amazon S3 で CloudFront を使用する最も簡単な方法は、Amazon S3 ですべてのオブジェクトを公開することです。そのためには、Amazon S3 にアップロードするオブジェクトごとに公開特権を明示的に有効にする必要があります。

コンテンツを公開しない場合は、CloudFront がコンテンツにアクセスできるように、CloudFront オリジンアクセスコントロール (OAC) を作成する必要があります。CloudFront オリジンアクセスコントロールの詳細については、「the section called “Amazon Simple Storage Service オリジンへのアクセスを制限する”」を参照してください。

オブジェクトのプロパティとバケットのプロパティはそれぞれ独立しています。権限は Amazon S3 のオブジェクトそれぞれに対して明示的に付与する必要があります。オブジェクトのプロパティはバケットから取得できないので、別途設定する必要があります。

代替ドメイン名 (CNAME) が正しく設定されていますか?

お使いのドメイン名に対応する CNAME レコードがすでに存在する場合は、そのレコードを更新または変更して、レコードがディストリビューションのドメイン名を指すようにしてください。

CNAME レコードが、Amazon S3 バケットではなく、ディストリビューションのドメイン名を指していることも確認してください。DNS システムの CNAME レコードがディストリビューションのドメイン名を指していることを確認することができます。そのためには、dig などの DNS ツールを使用します。

以下は、`images.example.com` というドメイン名に対する dig リクエストと、レスポンスのうち関連する部分のサンプルです。`ANSWER SECTION` の下で、`CNAME` が含まれる行を探します。CNAME の右側にある値が、CloudFront ディストリビューションのドメイン名であれば、ドメイン名の CNAME レコードが正しく設定されています。もしそれが Amazon S3 のオリジンサーバーのバケットや他のドメイン名になっている場合は、その CNAME レコードは正しく設定されていません。

```
[prompt]> dig images.example.com

; <<> DiG 9.3.3rc2 <<> images.example.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15917
;; flags: qr rd ra; QUERY: 1, ANSWER: 9, AUTHORITY: 2, ADDITIONAL: 0
;; QUESTION SECTION:
```

```
;images.example.com.   IN  A
;; ANSWER SECTION:
images.example.com. 10800 IN CNAME d111111abcdef8.cloudfront.net.
...
...
```

CNAME の詳細については、「代替ドメイン名 (CNAME) を追加することによって、カスタム URL を使用する」を参照してください。

参照している CloudFront ディストリビューションの URL は正しいですか?

参照している URL が、Amazon S3 バケットやカスタムオリジンではなく、CloudFront ディストリビューションのドメイン名 (または CNAME) を使用していることを確認してください。

カスタムオリジンに関するトラブルシューティングでサポートが必要ですか?

カスタムオリジンのトラブルシューティングで AWS のサポートが必要な場合は、おそらくリクエストからの `X-Amz-Cf-Id` ヘッダーエントリの調査が必要になります。現在ヘッダーエントリのログを記録していない場合は、将来に備えて記録することをお勧めします。詳しくは、「the section called “Amazon EC2 (または別のカスタムオリジン) を使用する”」を参照してください。詳細については、AWS サポートセンターまでお問い合わせください。

エラーメッセージ: Certificate: <certificate-id> is being used by CloudFront (証明書: <certificate-id> は CloudFront で使用されています)

問題: IAM 証明書ストアから SSL/TLS 証明書を削除しようとすると、メッセージ「Certificate: <certificate-id> is being used by CloudFront.」が表示されます。

解決方法: 各 CloudFront ディストリビューションは、デフォルトの CloudFront 証明書または独自 SSL/TLS 証明書のいずれかと関連付けられている必要があります。SSL/TLS 証明書を削除する前に、SSL/TLS 証明書を更新するか (現行の独自 SSL/TLS 証明書を別の独自 SSL/TLS 証明書に置き換える)、または使用する証明書を独自 SSL/TLS 証明書からデフォルトの CloudFront 証明書に戻してください。この問題を解決するには、次のいずれかの手順のステップを実行します。

- SSL/TLS 証明書をローテーションする
- カスタム SSL/TLS 証明書からデフォルト CloudFront 証明書に戻す

オリジンからのエラーレスポンスのトラブルシューティング

CloudFront からオリジンにオブジェクトをリクエストし、オリジンから HTTP 4xx または 5xx ステータスコードが返された場合は、CloudFront とオリジンとの間に通信の問題があります。以下のトピックでは、これらの HTTP ステータスコードの一般的な原因と、考えられる解決策について説明します。

トピック

HTTP 400 ステータスコード (Bad Request)

CloudFront ディストリビューションは、HTTP ステータスコード 400 Bad Request と以下のようなメッセージを含むエラーレスポンスを送信する場合があります。

The authorization header is malformed; the region '*<AWS Region>*' is wrong; expecting '*<AWS Region>*'

次に例を示します。

認証ヘッダーの形式が正しくありません。リージョン 'us-east-1' が間違っています。'us-west-2' を予期しています。

この問題は、次のシナリオで発生する可能性があります。

1. CloudFront ディストリビューションのオリジンが Amazon S3 バケットである。
2. S3 バケットを 1 つの AWS リージョンから別のリージョンに移動した。つまり、S3 バケットを削除し、後ほど同じバケット名で新しい S3 バケットを作成したが、元の S3 バケットがあった AWS リージョンとはちがうリージョンで作成したということです。

このエラーを修正するには、S3 バケットを現在の AWS リージョンで検索するように CloudFront ディストリビューションを更新します。

CloudFront ディストリビューションを更新するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudfront/v4/home で CloudFront コンソールを開きます。
2. このエラーを生成するディストリビューションを選択します。
3. [Origin and Origin Groups (オリジンおよびオリジングループ)] を選択します。
4. 移動した S3 バケットのオリジンを見つけます。このオリジンの横にあるチェックボックスをオンにして、[編集] を選択します。
5. [Yes, Edit (はい、編集します)] を選択します。[Yes, Edit (はい、編集します)] を選択する前に、設定を変更する必要はありません。

これらのステップを完了すると、CloudFront によりディストリビューションが再デプロイされます。ディストリビューションのデプロイ中は、[最終変更日] 列にステータスが [デプロイ中] と表示されます。デプロイの完了後、しばらくすると、`AuthorizationHeaderMalformed` エラーレスポンスを受信しなくなるはずです。

HTTP 502 ステータスコード (Bad Gateway)

HTTP 502 ステータスコード (Bad Gateway) は、CloudFront がオリジンサーバーに接続できず、リクエストされたオブジェクトを提供できなかった場合に返されます。

Lambda@Edge を使用している場合、問題は Lambda 検証エラーである可能性があります。HTTP 502 エラーとして `NonS3OriginDnsError` エラーコードが返された場合、DNS 設定の問題が原因で CloudFront がオリジンに接続できない可能性があります。

トピック

- CloudFront とカスタムオリジンサーバー間の SSL/TLS ネゴシエーションエラー
- サポートされている暗号化/プロトコルではオリジンが応答しません
- オリジンの SSL/TLS 証明書が期限切れ、無効、自己署名になっている、または間違った順番の証明書チェーンになっている
- オリジンがオリジン設定のポート指定に応答しません
- Lambda 検証エラー
- DNS エラー (NonS3OriginDnsError)

CloudFront とカスタムオリジンサーバー間の SSL/TLS ネゴシエーションエラー

カスタムオリジンを使用していて、CloudFront とオリジンとの間に HTTPS をリクエストするように CloudFront を設定した場合、問題はドメイン名の不一致であると考えられます。オリジンにインストールした SSL/TLS 証明書では、[Common Name (共通名)] フィールドにドメイン名が含まれ、[Subject Alternative Names (サブジェクトの代替名)] フィールドにもドメイン名がいくつか含まれることがあります。(CloudFront では証明書ドメイン名にワイルドカード文字を使用できます。) 証明書のドメイン名の 1 つは、次の値の 1 つまたは両方と一致する必要があります。

- ディストリビューションの該当するオリジンの [オリジンドメイン] に指定した値。
- `Host` ヘッダーをオリジンに転送するように CloudFront を設定した場合の `Host` ヘッダーの値。`Host` ヘッダーのオリジンへの転送の詳細については、「リクエストヘッダーに基づいてコンテンツをキャッシュする」を参照してください。

ドメイン名が一致しない場合、SSL/TLS ハンドシェイクは失敗し、CloudFront は HTTP 502 ステータスコード (Bad Gateway) を返し、`X-Cache` ヘッダーを `Error from cloudfront` に設定します。

証明書のドメイン名がディストリビューションまたは `Host` ヘッダーの [オリジンドメイン] と一致するかどうかを確認するには、オンライン SSL チェッカーまたは OpenSSL を使用できます。ドメイン名が一致しない場合、2 つのオプションがあります。

- ディストリビューションの該当するオリジンの [Origin Domain Name] に指定した値。
- `Host` ヘッダーをオリジンに転送するように CloudFront を設定した場合の `Host` ヘッダーの値。`Host` ヘッダーのオリジンへの転送の詳細については、「リクエストヘッダーに基づいてコンテンツをキャッシュする」を参照してください。

ドメイン名が一致しない場合、SSL/TLS ハンドシェイクは失敗し、CloudFront は HTTP 502 ステータスコード (Bad Gateway) を返し、`X-Cache` ヘッダーを `Error from cloudfront` に設定します。

証明書のドメイン名がディストリビューションまたは `Host` ヘッダーの [Origin Domain Name] と一致するかどうかを確認するには、オンライン SSL チェッカーまたは OpenSSL を使用できます。ドメイン名が一致しない場合、2 つのオプションがあります。

- 該当するドメイン名を含む新しい SSL/TLS 証明書を取得します。

AWS Certificate Manager (ACM) を使用する場合は、AWS Certificate Manager ユーザーガイドの「パブリック証明書をリクエストする」を参照して、新しい証明書をリクエストしてください。

- CloudFront で SSL を使用してオリジンに接続しないように、ディストリビューション設定を変更します。

オンライン SSL チェッカー

SSL テスト ツールを見つけるには、インターネットで「online ssl checker」を検索します。通常、ドメイン名を指定すると、ツールから SSL/TLS 証明書に関するさまざまな情報が返されます。証明書の [Common Names] フィールドまたは [Subject Alternative Names] フィールドにドメイン名が含まれていることを確認します。

OpenSSL

CloudFront からの HTTP 502 エラーをトラブルシューティングするには、OpenSSL を使用してオリジンサーバーへの SSL/TLS 接続を試行します。OpenSSL が接続できない場合、オリジンサーバーの SSL/TLS 設定に問題がある可能性があります。OpenSSL が接続を確立できる場合、証明書の共通名 (`Subject CN` フィールド) やサブジェクト代替名 (`Subject Alternative Name` フィールド) など、オリジンサーバーの証明書に関する情報を返します。

次の OpenSSL コマンドを使用して、オリジンサーバーへの接続をテストします (*`[########]`* を example.com などのオリジンサーバーのドメイン名に置き換えます)。

```
openssl s_client -connect origin domain name:443
```

次のことが当てはまるとします。

- オリジンサーバーは、複数の SSL/TLS 証明書を持つ複数のドメイン名をサポートしている
- `Host` ヘッダーをオリジンに転送するようにディストリビューションが設定されている

この場合、次の例のように OpenSSL コマンドに `-servername` オプションを追加します (*`CNAME`* をディストリビューションで設定した CNAME に置き換えます)。

```
openssl s_client -connect origin domain name:443 -servername CNAME
```

サポートされている暗号化/プロトコルではオリジンが応答しません

CloudFront は暗号とプロトコルを使用してオリジンサーバーに接続します。CloudFront がサポートする暗号とプロトコルのリストについては、「the section called “CloudFront とオリジンとの間

でサポートされているプロトコルと暗号”」を参照してください。オリジンが SSL/TLS 交換でこれらの暗号またはプロトコルに応答しない場合、CloudFront は接続を確立できません。SSL Labs などのオンラインツールを使って、オリジンが暗号とプロトコルをサポートすることを確認できます。[Host Name] フィールドでオリジンのドメイン名を入力し、[Submit] を選択します。テスト結果の [Common names] フィールドと [Alternative names] フィールドを見て、オリジンのドメイン名と一致しているかどうかを確認します。テスト完了後、テスト結果の [Protocols] または [Cipher Suites] セクションでオリジンがサポートする暗号とプロトコルを確認してください。それらを「the section called “CloudFront とオリジンとの間でサポートされているプロトコルと暗号”」のリストと比較します。

オリジンの SSL/TLS 証明書が期限切れ、無効、自己署名になっている、または間違った順番の証明書チェーンになっている

オリジンサーバーから、CloudFront が TCP 接続を中断する、HTTP ステータスコード 502 (Bad Gateway) を返す、`X-Cache` ヘッダーを `Error from cloudfront` に設定するなどのレスポンスがある場合:

- 証明書が期限切れです
- 証明書が無効です
- 証明書が自己署名です
- 間違った順番の証明書チェーンです

> **Note**
>
> 中間証明書を含む、証明書チェーンが完全でない場合も、CloudFront は TCP 接続を中断します。

カスタムオリジンサーバーで SSL/TLS 証明書をインストールする方法の詳細については、「the section called “カスタムオリジンに HTTPS を要求する”」を参照してください。

オリジンがオリジン設定のポート指定に応答しません

CloudFront ディストリビューションでオリジンを作成するときに、HTTP および HTTPS トラフィックのために CloudFront がオリジンへの接続に使用するポートを設定できます。デフォルトでは TCP 80/443です。これらのポートは変更可能です。これらのポートで、オリジンが何らかの理由でトラ

フィックを拒否している場合やバックエンドサーバーが応答していない場合、CloudFront は接続に失敗します。

これらの問題におけるトラブルシューティングには、インフラストラクチャで稼動するファイアウォールを確認し、サポートする IP 範囲がブロックされていないかを確認します。詳細については、Amazon Web Services 全般のリファレンス の AWS IP アドレスの範囲をご参照ください。ウェブサーバーがオリジンで稼働中であるかどうかも確認してください。

Lambda 検証エラー

Lambda@Edge を使用している場合、HTTP 502 ステータスコードは、Lambda 関数のレスポンスの形式が正しくないか、レスポンスに無効なコンテンツが含まれていたことを示している可能性があります。Lambda@Edge エラーのトラブルシューティングの詳細については、「Lambda@Edge 関数をテストおよびデバッグする」を参照してください。

DNS エラー (**`NonS3OriginDnsError`**)

`NonS3OriginDnsError` エラーコードを含む HTTP 502 エラーは、CloudFront がオリジンに接続できないという、DNS 設定の問題があることを示しています。このエラーが CloudFront で発生した場合は、オリジンの DNS 設定が正常に機能していることを確認してください。

CloudFront は、期限切れのオブジェクトや、キャッシュに保存されていないオブジェクトをリクエストされると、オリジンにリクエストしてオブジェクトを取得しようとします。オリジンに対して正常なリクエストを行うため、CloudFront はオリジンドメインで DNS 解決を実行します。ドメインの DNS サービスで問題が発生している場合、CloudFront はドメイン名を解決して IP アドレスを取得できないため、HTTP 502 エラー (`NonS3OriginDnsError`) が発生します。この問題を解決するには、DNS プロバイダーにお問い合わせください。Amazon Route 53 を使用している場合は、「Route 53 DNS サービスを使用している自分のウェブサイトにアクセスできないのはなぜですか?」を参照してください。

この問題の詳しいトラブルシューティングを行うには、オリジンのルートドメインまたは zone apex (`example.com` など) の権威ネームサーバーが正しく機能していることを確認します。dig や nslookup などのツールにより、次のコマンドを使用して apex オリジンのネームサーバーを検索できます。

```
dig OriginAPEXDomainName NS +short
```

```
nslookup -query=NS OriginAPEXDomainName
```

ネームサーバーの名前がある場合、次のコマンドを使用して、それらに対してオリジンのドメイン名のクエリを実行し、各サーバーが応答して答えを返すことを確認します。

```
dig OriginDomainName @NameServer
```

```
nslookup OriginDomainName NameServer
```

> **Important**
>
> この DNS トラブルシューティングは、公共のインターネットに接続しているコンピュータを使用して実行してください。CloudFront はインターネット上のパブリック DNS を使用してオリジンドメインを解決するため、同様の状況でトラブルシューティングを行うことが重要です。

オリジンがサブドメインであり、このサブドメインの DNS 権限がルートドメインとは異なるネームサーバーに委任されている場合は、ネームサーバー (NS) および Start of Authority (SOA) レコードが、このサブドメインに対して正しく設定されていることを確認してください。これらのレコードは、前述の例と同様のコマンドを使用して確認できます。

DNS の詳細については、Amazon Route 53 ドキュメントの「ドメインネームシステム (DNS) の概念」を参照してください。

HTTP 503 ステータスコード (Service Unavailable)

通常、HTTP 503 ステータスコード (Service Unavailable) は、オリジンサーバーのパフォーマンスの問題を示します。まれに、エッジロケーションでリソースが制限されているため、CloudFront が一時的にリクエストを満たせないことを示している場合もあります。

Lambda@Edge または CloudFront Functions を使用している場合、実行エラーまたは Lambda@Edge の制限超過エラーが原因である可能性があります。

トピック

- オリジンサーバーにリクエスト率をサポートする十分な容量がない
- エッジロケーションのリソースが制限されているために CloudFront でエラーが発生した
- Lambda@Edge または CloudFront Functions の実行エラー
- Lambda@Edge の制限超過

オリジンサーバーにリクエスト率をサポートする十分な容量がない

オリジンサーバーが利用できないか、受信リクエストを処理できない場合は、HTTP 503 ステータスコード (Service Unavailable) が返されます。この場合、CloudFront はエラーをユーザーに中継します。この問題を解決するには、以下の手順をお試しください。

- Amazon S3 をオリジンサーバーとして使用している場合:
  - パーティショニングされた Amazon S3 プレフィックスごとに毎秒 3,500 件の PUT/COPY/POST/DELETE リクエストまたは 5,500 件の GET/HEAD リクエストを送信できます。Amazon S3 から 503 Slow Down レスポンスが返された場合は、通常、特定の Amazon S3 プレフィックスに対するリクエストレートが過剰であることを示しています。

    リクエストレートは S3 バケットのプレフィックスごとに適用されるため、オブジェクトは複数のプレフィックスに分散する必要があります。プレフィックスに対するリクエストレートが徐々に増えると、Amazon S3 はスケールアップして各プレフィックスのリクエストを個別に処理します。その結果、バケットが処理する全体的なリクエストレートは、プレフィックス数の倍数になります。
  - 詳細については、Amazon Simple Storage Service ユーザーガイドの「Amazon S3 のパフォーマンスの最適化」を参照してください。
- Elastic Load Balancing をオリジンサーバーとして使用している場合:
  - バックエンドインスタンスがヘルスチェックに応答できることを確認します。
  - ロードバランサーとバックエンドインスタンスが負荷を処理できることを確認します。

  詳細については、以下を参照してください。
  - Classic Load Balancer の使用中に返される 503 エラーをトラブルシューティングするにはどうしたらよいですか?
  - Application Load Balancer からの 503 (サービス利用不可) エラーのトラブルシューティング方法を教えてください。
- カスタムオリジンを使用している場合:
  - アプリケーションログを調べて、オリジンのメモリ、CPU、ディスクサイズなどのリソースが十分であることを確認します。
  - Amazon EC2 をバックエンドとして使用している場合は、受信されるリクエストを満たす適切なリソースがインスタンスタイプにあることを確認します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスタイプ」を参照してください。
- API Gateway を使用している場合:

- API Gateway API がレスポンスを受信できない場合、このエラーはバックエンド統合に関連しています。バックエンドサーバーが次の状態である可能性があります。
  - 容量を超えて過負荷状態であり、新しいクライアントリクエストを処理できません。
  - 一時的にメンテナンス中です。
- このエラーを解決するには、API Gateway アプリケーションログを調べて、バックエンドの容量、統合、その他に問題があるかどうかを確認します。

エッジロケーションのリソースが制限されているために CloudFront でエラーが発生した

このエラーは、CloudFront から次の最も利用可能なエッジロケーションにリクエストをルーティングできず、リクエストを満たすことができないという、まれな状況で発生します。このエラーは CloudFront ディストリビューションで負荷テストを実行するときによく発生します。これを回避するには、「the section called “CloudFront の負荷テスト”」のガイドラインに従って 503 (キャパシティー超過) エラーが発生しないようにします。

このエラーが本稼働環境で発生した場合は、AWS Supportまでお問い合わせください。

Lambda@Edge または CloudFront Functions の実行エラー

Lambda@Edge または CloudFront Functions を使用している場合、HTTP 503 ステータスコードは、関数が実行エラーを返したことを示している可能性があります。

Lambda@Edge エラーを特定して解決する方法の詳細については、「Lambda@Edge 関数をテストおよびデバッグする」を参照してください。

CloudFront Functions のテストの詳細については、「関数をテストする」を参照してください。

Lambda@Edge の制限超過

Lambda@Edge を使用している場合、HTTP 503 ステータスコードは、Lambda がエラーを返したことを示している可能性があります。このエラーは、以下のいずれかが原因である可能性があります。

- 関数の実行数が、AWS リージョンでの実行をスロットリングするために Lambda が設定したクォータ (同時実行数または呼び出し頻度) の 1 つを超えている。
- 関数が Lambda 関数のタイムアウトクォータを超過している。

Lambda@Edge のクォータの詳細については、「Lambda@Edge のクォータ」を参照してください。Lambda@Edge エラーを特定して解決する方法の詳細については、「the section called “テストおよびデバッグする”」を参照してください。「AWS Lambda 開発者ガイド」の「Lambda サービスクォータ」を参照することもできます。

HTTP 504 ステータスコード (Gateway Timeout)

HTTP 504 ステータスコード (Gateway Timeout) は、CloudFront がオリジンにリクエストを転送したとき (リクエストされたオブジェクトがエッジキャッシュに存在しなかったため) に、以下のいずれかの状況が発生したことを示しています。

- オリジンが HTTP 504 ステータスコードを CloudFront に返した。
- リクエストの期限切れまでにオリジンが応答しなかった。

トラフィックがファイアウォールまたはセキュリティグループによってオリジンからブロックされているか、インターネットでオリジンにアクセスできない場合、CloudFront は HTTP 504 ステータスコードを返します。最初に、これらの問題を確認します。次に、アクセスに問題がない場合は、アプリケーションの遅延とサーバーのタイムアウトを調べると、問題の特定と修正に役立ちます。

トピック

- CloudFront トラフィックを許可するようにオリジンサーバーのファイアウォールを設定する
- CloudFront トラフィックを許可するようにオリジンサーバーのセキュリティグループを設定する
- インターネットでカスタムオリジンサーバーをアクセス可能にする
- オリジンサーバーでアプリケーションからの遅延したレスポンスを見つけて修正する

CloudFront トラフィックを許可するようにオリジンサーバーのファイアウォールを設定する

オリジンサーバーのファイアウォールが CloudFront トラフィックをブロックしている場合、CloudFront は HTTP 504 ステータスコードを返すため、これが問題でないことを確認した上で、他の問題を確認できます。

これがファイアウォールの問題であるかどうかを判断するために使用する方法は、オリジンサーバーが使用しているシステムによって異なります。

- Linux サーバーで IPTable ファイアウォールを使用している場合は、IPTables を操作するのに役立つツールと情報を検索できます。

- Windows サーバーで Windows ファイアウォールを使用している場合は、Microsoft ドキュメントの「Add or Edit Firewall Rule」(ファイアウォール規則を追加または編集する) を参照してください。

オリジンサーバーでファイアウォール設定を評価するときは、公開されている IP アドレス範囲に基づいて、CloudFront エッジロケーションからのトラフィックをブロックしているファイアウォールまたはセキュリティルールを探します。詳細については、「CloudFront エッジサーバーの場所と IP アドレス範囲」を参照してください。

CloudFront の IP アドレス範囲がオリジンサーバーへの接続を許可されている場合は、サーバーのセキュリティルールを更新して変更を反映します。Amazon SNS トピックにサブスクライブして、IP アドレス範囲ファイルが更新されたときに通知を受け取ることができます。通知を受け取ったら、コードを使用してファイルを取得し、解析して、ローカル環境を調整することができます。詳細については、AWS ニュースブログの「Subscribe to AWS Public IP Address Changes via Amazon SNS」を参照してください。

CloudFront トラフィックを許可するようにオリジンサーバーのセキュリティグループを設定する

オリジンで Elastic Load Balancing を使用している場合は、ELB セキュリティグループを確認し、セキュリティグループで CloudFront からのインバウンドトラフィックを許可していることを確認します。

また、AWS Lambda を使用してセキュリティグループを自動的に更新することで、CloudFront からのインバウンドトラフィックを許可することもできます。

インターネットでカスタムオリジンサーバーをアクセス可能にする

カスタムオリジンサーバーがインターネットで公開されておらず、CloudFront からアクセスできない場合は、HTTP 504 エラーが返されます。

CloudFront エッジロケーションはインターネットを介してオリジンサーバーに接続します。カスタムオリジンがプライベートネットワークにある場合、CloudFront はオリジンに到達できません。このため、内部の Classic Load Balancer などのプライベートサーバーをオリジンサーバーとして CloudFront で使用することはできません。

インターネットトラフィックがオリジンサーバーに接続できることを確認するには、次のコマンドを実行します (*OriginDomainName* はサーバーのドメイン名です)。

HTTPS トラフィックの場合:

- nc -zv *OriginDomainName* 443
- telnet *OriginDomainName* 443

HTTP トラフィックの場合:

- nc -zv *OriginDomainName* 80
- telnet *OriginDomainName* 80

オリジンサーバーでアプリケーションからの遅延したレスポンスを見つけて修正する

サーバーのタイムアウトは、多くの場合、アプリケーションの応答に非常に長い時間がかかっているか、タイムアウト値の設定が低すぎる場合に発生します。

HTTP 504 エラーを手早く修正するには、ディストリビューションの CloudFront タイムアウト値を高く設定します。ただし、アプリケーションとオリジンサーバーのパフォーマンスとレイテンシーの問題があれば、最初にその問題に対応することをお勧めします。次に、HTTP 504 エラーを回避してユーザーに良好な応答性を提供する、適切なタイムアウト値を設定できます。

パフォーマンスの問題を見つけて修正するための手順について、以下に概要を示します。

1. ウェブアプリケーションの一般的な高負荷のレイテンシー (応答性) を測定します。
2. 必要に応じて CPU やメモリなどのリソースを追加します。データベースクエリを高負荷シナリオに対応するようにチューニングするなど、問題に対応する他のステップを実行します。
3. 必要に応じて、CloudFront ディストリビューションのタイムアウト値を調整します。

各ステップの詳細を以下に示します。

一般的な高負荷のレイテンシーの測定

1 台以上のバックエンドウェブアプリケーションサーバーで長いレイテンシーが発生しているかどうか調べるには、各サーバーで次の Linux curl コマンドを実行します。

```
curl -w "Connect time: %{time_connect} Time to first byte: %{time_starttransfer} Total
 time: %{time_total} \n" -o /dev/null https://www.example.com/yourobject
```

> **Note**
>
> サーバーで Windows を実行する場合は、Windows 用の curl を検索およびダウンロードして、類似したコマンドを実行できます。

サーバーで実行するアプリケーションのレイテンシーを測定および評価する場合は、次の点に留意します。

- レイテンシーの値は、各アプリケーションに対して相対的です。ただし、先頭バイトまでの時間は、秒単位またはそれ以上ではなく、ミリ秒単位が合理的です。
- 通常の負荷でアプリケーションのレイテンシーを測定して問題がなくても、高負荷がかかった場合に、ビューワーにタイムアウトが発生する可能性があることに注意してください。需要が高い場合、サーバーでレスポンスの遅延が発生するか、まったく応答しないことがあります。高負荷に伴うレイテンシーの問題を避けるために、CPU、メモリ、ディスクの読み取りと書き込みなど、サーバーのリソースをチェックし、サーバーが高負荷に合わせてスケールできることを確認します。

  次の Linux コマンドを実行して、Apache プロセスによって使用されているメモリを確認できます。

  ```
  watch -n 1 "echo -n 'Apache Processes: ' && ps -C apache2 --no-headers |
  wc -l && free -m"
  ```
- サーバーでの高い CPU 使用率により、アプリケーションのパフォーマンスが大幅に低下する場合があります。Amazon EC2 インスタンスをバックエンドサーバーとして使用している場合は、サーバーの CloudWatch メトリクスで CPU 使用率を確認します。詳細については、Amazon CloudWatch ユーザーガイドを参照してください。または、独自のサーバーを使用している場合は、CPU 使用率を確認する方法について、サーバーのヘルプドキュメントを参照してください。
- リクエストの量が多くてデータベースクエリが遅くなるなど、高負荷に伴って発生する可能性がある他の問題を確認します。

リソースの追加およびサーバーとデータベースのチューニング

アプリケーションとサーバーの応答性を評価したら、一般的なトラフィックと高負荷の状況に対する十分なリソースがあることを確認します。

- 独自のサーバーがある場合は、評価に基づいて、ビューワーリクエストを処理する十分な CPU、メモリ、およびディスクスペースがあることを確認します。
- Amazon EC2 インスタンスをバックエンドサーバーとして使用している場合は、受信されるリクエストを満たす適切なリソースがインスタンスタイプにあることを確認します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスタイプ」を参照してください。

さらに、タイムアウトを避けるために次のチューニングステップを検討します。

- curl コマンドによって返される先頭バイトまでの時間の値が高いと思われる場合は、アプリケーションのパフォーマンスを向上させるステップを実行します。アプリケーションの応答性の向上は、タイムアウトエラーを減らすうえで有効です。
- データベースクエリをチューニングし、パフォーマンスを低下させることなく高いリクエストボリュームを処理できるようにします。
- バックエンドサーバーでキープアライブ (持続的) 接続を設定します。このオプションは、それ以降のリクエストまたはユーザーに対して接続を再確立する必要があるときに発生するレイテンシーを回避するために有効です。
- ELB をオリジンとして使用している場合、レイテンシーを減らす方法については、ナレッジセンター記事「ELB Classic Load Balancer のレイテンシーが高い場合のトラブルシューティング方法を教えてください」の推奨事項を参照してください。

必要に応じて、CloudFront タイムアウト値を調整する

アプリケーションの低いパフォーマンス、オリジンサーバーの容量、およびその他の問題について評価、対応したが、それでもビューワーに HTTP 504 エラーが発生する場合は、オリジン応答タイムアウトに対してディストリビューションで指定されている時間の変更を検討します。詳細については、「the section called “応答タイムアウト (カスタムオリジンのみ)”」を参照してください。

CloudFront の負荷テスト

従来の負荷テストの方法は CloudFront では適切に機能しません。CloudFront が、DNS を使用して、地理的に分散したエッジロケーション間および各エッジロケーション内で負荷を分散するためです。クライアントが CloudFront にコンテンツをリクエストすると、クライアントは一連の IP アドレスが含まれた DNS レスポンスを受け取ります。DNS が返す IP アドレスの 1 つだけにリクエストを送信してテストするなら、1 つの CloudFront エッジロケーション内のリソースのごく一部分にのみテストを実行することなり、実際のトラフィックパターンを正確には表しません。この方法でテス

トした場合、リクエストするデータの量によっては、テスト対象になったごく一部分の CloudFront サーバーに負荷がかかりすぎて、パフォーマンスが低下することもあります。

CloudFront は、複数の地理的リージョンにわたってクライアント IP アドレスや DNS リゾルバーの異なるビューワーに応じて、スケーリングされるように設計されています。CloudFront のパフォーマンスを正確に評価する負荷テストを実行するには、以下の手順をすべて実行することをお勧めします。

- 複数の地理的リージョンからクライアントのリクエストを送信します。
- 各クライアントが独立した DNS リクエストを行うようにテストを設定します。結果として、クライアントごとに異なる IP アドレスのセットを DNS から受け取ります。
- リクエストを行うクライアントごとに、DNS から返された IP アドレスのセット全体にクライアントリクエストを分散します。これにより、CloudFront エッジロケーション内の複数のサーバーに負荷が分散されます。

> **メモ**
>
> - Lambda@Edge ビューワーリクエストまたはビューワーレスポンストリガーがあるキャッシュ動作に対しては負荷テストを実行できません。
> - オリジンシールドが有効になっているオリジンに対しては負荷テストを実行できません。

クォータ

CloudFront には、次のクォータが適用されます。

トピック

一般的なクォータ

エンティティ	デフォルトのクォータ
ディストリビューションごとのデータ転送レート	150 Gbps クォータ引き上げのリクエスト

エンティティ	デフォルトのクォータ
1 秒あたり、ディストリビューションあたりのリクエスト	250,000 クォータ引き上げのリクエスト
ディストリビューションに追加できるタグ	50 クォータ引き上げのリクエスト
ディストリビューションごとに配信可能なファイル数	クォータなし
ヘッダーとクエリ文字列は含むが、本文のコンテンツは含まないリクエストやオリジンレスポンスの最大長	20,480 バイト
URL の最大長	8,192 バイト

ディストリビューションの一般的なクォータ

エンティティ	デフォルトのクォータ
ディストリビューションあたりの代替ドメイン名 (CNAME) 詳細については、「代替ドメイン名 (CNAME) を追加することによって、カスタム URL を使用する」を参照してください。	100 クォータ引き上げのリクエスト
ディストリビューションあたりのキャッシュ動作	25 クォータ引き上げのリクエスト
オリジン別の接続試行回数 詳細については、「接続の試行」を参照してください。	1 ~ 3

エンティティ	デフォルトのクォータ
オリジン別の接続タイムアウト 詳細については、「接続タイムアウト」を参照してください。	1 ~ 10 秒
AWS アカウントあたりのディストリビューション 詳細については、「ディストリビューションを作成する」を参照してください。	200 クォータ引き上げのリクエスト
オリジンアクセスコントロールあたりのディストリビューション数	100 クォータ引き上げのリクエスト
ファイル圧縮: CloudFront が圧縮するファイルサイズの範囲 詳細については、「圧縮ファイルを供給する」を参照してください。	1,000 ~ 10,000,000 バイト
オリジンあたりのキープアライブタイムアウト 詳細については、「キープアライブタイムアウト (カスタムオリジンのみ)」を参照してください。	1 ~ 60 秒 クォータ引き上げのリクエスト
HTTP GET レスポンスあたりのキャッシュ可能な最大ファイルサイズ。 HTTP GET のレスポンスのみがキャッシュされます。POST または PUT のレスポンスはキャッシュされません。	50 GB
AWS アカウントあたりのオリジンアクセスコントロール	100
AWS アカウントあたりのオリジンアクセスアイデンティティ	100 クォータ引き上げのリクエスト

エンティティ	デフォルトのクォータ
ディストリビューションあたりのオリジン	25 クォータ引き上げのリクエスト
ディストリビューションあたりのオリジングループ	10 クォータ引き上げのリクエスト
オリジン別の応答タイムアウト 詳細については、「応答タイムアウト (カスタムオリジンのみ)」を参照してください。	1 ~ 60 秒 クォータ引き上げのリクエスト
AWS アカウントあたりのステージングディストリビューション 詳細については、「the section called “継続的デプロイを使用して変更を安全にテストする”」を参照してください。	20 クォータ引き上げのリクエスト

ポリシーの一般的なクォータ

エンティティ	デフォルトのクォータ
AWS アカウントあたりのキャッシュポリシー	20 クォータ引き上げのリクエスト
同じキャッシュポリシーに関連付けられたディストリビューション	100
キャッシュポリシーあたりのクエリ文字列	10 クォータ引き上げのリクエスト

エンティティ	デフォルトのクォータ
キャッシュポリシーあたりのヘッダー	10 クォータ引き上げのリクエスト
キャッシュポリシーあたりの Cookie	10 クォータ引き上げのリクエスト
キャッシュポリシー内のすべてのクエリ文字列、ヘッダー、および Cookie 名の合計長	1024
AWS アカウントあたりのオリジンリクエストポリシー	20 クォータ引き上げのリクエスト
同じオリジンリクエストポリシーに関連付けられたディストリビューション	100
オリジンリクエストポリシーあたりのクエリ文字列	10 クォータ引き上げのリクエスト
オリジンリクエストポリシーあたりのヘッダー	10 クォータ引き上げのリクエスト
オリジンリクエストポリシーあたりの Cookie	10 クォータ引き上げのリクエスト
キャッシュリクエストポリシー内のすべてのクエリ文字列、ヘッダー、および Cookie 名の合計長	1024

エンティティ	デフォルトのクォータ
AWS アカウントあたりのレスポンスヘッダーポリシー	20 クォータ引き上げのリクエスト
同じレスポンスヘッダーポリシーに関連付けられたディストリビューション	100 クォータ引き上げのリクエスト
レスポンスヘッダーポリシーごとのカスタムヘッダー	10 クォータ引き上げのリクエスト
AWS アカウントあたりの継続的デプロイポリシー	20 クォータ引き上げのリクエスト

CloudFront Functions のクォータ

エンティティ	デフォルトのクォータ
AWS アカウント あたりの関数	100
最大関数サイズ	10 KB クォータ引き上げのリクエスト
最大関数メモリ	2 MB
同じ関数に関連付けられたディストリビューション	100

これらのクォータに加えて、CloudFront Functions を使用する場合は、他にもいくつかの制限があります。詳細については、「CloudFront Functions に対する制限」を参照してください。

キー値ストアのクォータ

エンティティ	デフォルトのクォータ
キーと値のペアにおけるキーの最大サイズ	512 バイト
キーと値のペアにおける値の最大サイズ	1 KB
1 回の API リクエストで更新できるキー値ペアの最大数	50 個のキーまたは 3 MB のペイロード (どちらか先に達した方)
個々のキーと値のストアの最大サイズ	5 MB
1 つのキー値ストアに関連付けることができる関数の最大数	10
関数ごとのキー値ストアの最大数	1
アカウントごとのキー値ストアの最大数	50 クォータ引き上げのリクエスト

Lambda@Edge のクォータ

このセクションのクォータは、Lambda@Edge に適用されます。これらのクォータは、同じく適用されるデフォルトの AWS Lambda クォータへの追加となります。Lambda クォータについては、AWS Lambda デベロッパーガイドの「クォータ」を参照してください。

> **Note**
>
> Lambda は、AWS アカウントのクォータ内で、トラフィックの増加に応じて容量を動的にスケールします。詳細については、AWS Lambda デベロッパーガイドの「関数スケーリング」を参照してください。

一般的なクォータ

エンティティ	デフォルトのクォータ
Lambda@Edge 関数を持つことができる AWS アカウントあたりのディストリビューション	500 クォータ引き上げのリクエスト
ディストリビューションごとの Lambda@Edge 関数	100 クォータ引き上げのリクエスト
1 秒あたりのリクエスト	10,000 (各 AWS リージョン 内) クォータ引き上げのリクエスト
同時実行数 詳細については、AWS Lambda デベロッパーガイドの「関数スケーリング」を参照してください。	1,000 (各AWS リージョン 内) クォータ引き上げのリクエスト
同じ関数に関連付けられたディストリビューション	500

イベントタイプによって異なるクォータ

エンティティ	ビューワーリクエストイベントとビューワーレスポンスイベント	オリジンリクエストイベントとオリジンレスポンスイベント
関数のメモリサイズ	128 MB	Lambda のクォータと同じ

エンティティ	ビューワーリクエストイベントとビューワーレスポンスイベント	オリジンリクエストイベントとオリジンレスポンスイベント
関数タイムアウト。関数は、AWS リージョン内の Amazon S3 バケット、DynamoDB テーブル、または Amazon EC2 インスタンスなどのリソースに対してネットワークコールを実行できます。	5 秒	30 秒
ヘッダーと本文を含む、Lambda 関数によって生成されたレスポンスのサイズ	40 KB	1 MB
Lambda 関数および組み込みライブラリの最大圧縮サイズ	1 MB	50 MB

これらのクォータに加えて、Lambda@Edge 関数を使用する場合は、他にもいくつかの制限があります。詳細については、「Lambda@Edge に対する制限」を参照してください。

SSL 証明書のクォータ

エンティティ	デフォルトのクォータ
専用 IP アドレスを使用して HTTPS リクエストに対応する際の AWS アカウントあたりの SSL 証明書数 (SNI を使用して HTTPS リクエストに対応する際はクォータなし) 詳細については、「CloudFront で HTTPS を使用する」を参照してください。	2 クォータ引き上げのリクエスト
CloudFront ディストリビューションに関連付けることができる SSL 証明書	1

SSL 証明書がビューワーと CloudFront の間の HTTPS 通信専用であり、AWS Certificate Manager (ACM) または IAM 証明書ストアを使用して証明書をプロビジョニングまたはインポートした場合は、追加のクォータが適用されます。詳細については、「CloudFront で SSL/TLS 証明書を使用する場合のクォータ (ビューワーと CloudFront との間の HTTPS のみ)」を参照してください。

AWS Certificate Manager (ACM) にインポートしたり、AWS Identity and Access Management (IAM) にアップロードしたりできる SSL 証明書の数にもクォータがあります。詳細については、「SSL/TLS 証明書のクォータを引き上げる」を参照してください。

無効化のクォータ

エンティティ	デフォルトのクォータ
ファイルの無効化: ワイルドカードの無効化を除く、アクティブな無効化リクエストで許可されるファイルの最大数 詳細については、「ファイルを無効化してコンテンツを削除する」を参照してください。	3,000
ファイルの無効化: 許可されるアクティブなワイルドカード無効化の最大数	15
ファイルの無効化: 1 つのワイルドカードの無効化で処理できるファイルの最大数	クォータなし

キーグループのクォータ

エンティティ	デフォルトのクォータ
1 つのキーグループのパブリックキー	5 クォータ引き上げのリクエスト
1 つのキャッシュ動作に関連付けられたキーグループ	4

エンティティ	デフォルトのクォータ
	クォータ引き上げのリクエスト
AWS アカウントあたりのキーグループ数	10 クォータ引き上げのリクエスト
1 つのキーグループに関連付けられたディストリビューション	100 クォータ引き上げのリクエスト

WebSocket 接続のクォータ

エンティティ	デフォルトのクォータ
オリジン応答タイムアウト (アイドルタイムアウト)	10 分 CloudFront が、過去 10 分以内にオリジンからクライアントに送信されたバイトを検出しなかった場合、接続はアイドル状態であると見なされ、閉じられます。

フィールドレベル暗号化のクォータ

エンティティ	デフォルトのクォータ
暗号化するフィールドの最大長	16 KB

エンティティ	デフォルトのクォータ
詳細については、「フィールドレベル暗号化を使用した機密データの保護」を参照してください。	
フィールドレベル暗号化が設定されている場合のリクエストボディの最大フィールド数	10
フィールドレベル暗号化が設定されている場合のリクエストボディの最大長	1 MB
1 つの AWS アカウントに関連付けることができるフィールドレベル暗号化設定の最大数	10
1 つの AWS アカウントに関連付けることができるフィールドレベル暗号化プロファイルの最大数	10
1 つの AWS アカウントに追加できるパブリックキーの最大数	10
1 つのプロファイルに指定できる暗号化するフィールドの最大数	10
1 つのフィールドレベル暗号化設定に関連付けることができる CloudFront ディストリビューションの最大数	20
フィールドレベル暗号化設定に含めることができるクエリ引数プロファイルマッピングの最大数	5

Cookie のクォータ (従来のキャッシュ設定)

これらのクォータは、CloudFront の従来のキャッシュ設定に適用されます。従来の設定の代わりに、キャッシュポリシーまたはオリジンリクエストポリシーを使用することをお勧めします。

エンティティ	デフォルトのクォータ
キャッシュ動作あたりの Cookie	10

エンティティ	デフォルトのクォータ
詳細については、「Cookie に基づいてコンテンツをキャッシュする」を参照してください。	クォータ引き上げのリクエスト
Cookie 名の総バイト数 (すべての Cookie をオリジンに転送するように CloudFront を設定している場合は適用されません)	512 から Cookie の数を引いたもの

クエリ文字列のクォータ (従来のキャッシュ設定)

これらのクォータは、CloudFront の従来のキャッシュ設定に適用されます。従来の設定の代わりに、キャッシュポリシーまたはオリジンリクエストポリシーを使用することをお勧めします。

エンティティ	デフォルトのクォータ
クエリ文字列の最大文字数	128 文字
同じパラメータ内のすべてのクエリ文字列の最大合計文字数	512 文字
キャッシュ動作あたりのクエリ文字列 詳細については、「クエリ文字列パラメータに基づいてコンテンツをキャッシュする」を参照してください。	10 クォータ引き上げのリクエスト

ヘッダーのクォータ

エンティティ	デフォルトのクォータ
キャッシュ動作ごとのヘッダー (従来のキャッシュ設定) 詳細については、「the section called “リクエストヘッダーに基づいてコンテンツをキャッシュする”」を参照してください。	10 クォータ引き上げのリクエスト

エンティティ	デフォルトのクォータ
カスタムヘッダーあり: オリジンリクエストを追加するように CloudFront を設定可能なカスタムヘッダーの最大数 詳細については、「the section called “オリジンリクエストにカスタムヘッダーを追加する”」を参照してください。	10 クォータ引き上げのリクエスト
カスタムヘッダー: レスポンスヘッダーポリシーに追加できるカスタムヘッダーの最大数	10 クォータ引き上げのリクエスト
カスタムヘッダーあり: ヘッダー名の最大長	256 文字
カスタムヘッダー: ヘッダー値の最大長	1,783 文字
カスタムヘッダー: 結合されるすべてのヘッダー値および名前の最大長	10,240 文字
`Content-Security-Policy` ヘッダーの値の最大長	1,783 文字 クォータ引き上げのリクエスト

AWS SDK を使用した CloudFront のコードの例

次のコード例は、AWS Software Development Kit (SDK) で CloudFront を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- AWS SDK を使用した CloudFront のアクション
  - AWS SDK または CLI で CreateDistribution を使用する
  - AWS SDK または CLI で CreateFunction を使用する
  - AWS SDK または CLI で CreateInvalidation を使用する
  - AWS SDK または CLI で CreateKeyGroup を使用する
  - AWS SDK または CLI で CreatePublicKey を使用する
  - AWS SDK または CLI で DeleteDistribution を使用する
  - AWS SDK または CLI で GetCloudFrontOriginAccessIdentity を使用する
  - AWS SDK または CLI で GetCloudFrontOriginAccessIdentityConfig を使用する
  - AWS SDK または CLI で GetDistribution を使用する
  - AWS SDK または CLI で GetDistributionConfig を使用する
  - AWS SDK または CLI で ListCloudFrontOriginAccessIdentities を使用する
  - AWS SDK または CLI で ListDistributions を使用する
  - AWS SDK または CLI で UpdateDistribution を使用する
- SDK AWS を使用する CloudFront のシナリオ
  - SDKAWS を使用して CloudFront 署名リソースを削除する
  - AWS SDK を使用して署名付き URL と Cookie を作成する

AWS SDK を使用した CloudFront のアクション

次のコード例は、AWS SDK を使用して個々の CloudFront アクションを実行する方法を示しています。これらは、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋であり、CloudFront API を呼び出すために使用します。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細な一覧については、「Amazon CloudFront API リファレンス」を参照してください。

例

- AWS SDK または CLI で CreateDistribution を使用する
- AWS SDK または CLI で CreateFunction を使用する
- AWS SDK または CLI で CreateInvalidation を使用する
- AWS SDK または CLI で CreateKeyGroup を使用する
- AWS SDK または CLI で CreatePublicKey を使用する
- AWS SDK または CLI で DeleteDistribution を使用する
- AWS SDK または CLI で GetCloudFrontOriginAccessIdentity を使用する
- AWS SDK または CLI で GetCloudFrontOriginAccessIdentityConfig を使用する
- AWS SDK または CLI で GetDistribution を使用する
- AWS SDK または CLI で GetDistributionConfig を使用する
- AWS SDK または CLI で ListCloudFrontOriginAccessIdentities を使用する
- AWS SDK または CLI で ListDistributions を使用する
- AWS SDK または CLI で UpdateDistribution を使用する

AWS SDK または CLI で `CreateDistribution` を使用する

以下のコード例は、`CreateDistribution` の使用方法を示しています。

CLI

AWS CLI

CloudFront ディストリビューションを作成するには

以下の例では、awsexamplebucket という名前の S3 バケットのディストリビューションを作成し、コマンドライン引数を使用してデフォルトのルートオブジェクトとして index.html を指定しています。

```
aws cloudfront create-distribution \
    --origin-domain-name awsexamplebucket.s3.amazonaws.com \
    --default-root-object index.html
```

次の例に示すように、コマンドライン引数を使用する代わりに、JSON ファイルでディストリビューション設定を指定できます。

```
aws cloudfront create-distribution \
    --distribution-config file://dist-config.json
```

dist-config.json ファイルは、以下を含む現在のフォルダ内にある JSON ドキュメントです。

```
{
    "CallerReference": "cli-example",
    "Aliases": {
        "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
                "DomainName": "awsexamplebucket.s3.amazonaws.com",
                "OriginPath": "",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "OriginGroups": {
        "Quantity": 0
    },
```

```
    "DefaultCacheBehavior": {
        "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
            },
            "Headers": {
                "Quantity": 0
            },
            "QueryStringCacheKeys": {
                "Quantity": 0
            }
        },
        "TrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ViewerProtocolPolicy": "allow-all",
        "MinTTL": 0,
        "AllowedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ]
            }
        },
        "SmoothStreaming": false,
        "DefaultTTL": 86400,
        "MaxTTL": 31536000,
        "Compress": false,
        "LambdaFunctionAssociations": {
            "Quantity": 0
        },
        "FieldLevelEncryptionId": ""
    },
    "CacheBehaviors": {
```

```
        "Quantity": 0
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Comment": "",
    "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
}
```

ディストリビューション情報をコマンドライン引数で指定する場合も、JSON ファイルで指定する場合も、出力は変わりません。

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EMLARXS9EXAMPLE",
    "ETag": "E9LHASXEXAMPLE",
    "Distribution": {
        "Id": "EMLARXS9EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-11-22T00:55:15.705Z",
        "InProgressInvalidationBatches": 0,
```

```
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
                        "DomainName": "awsexamplebucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-
example",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
```

```
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
```

```
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「CreateDistribution」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

次の例では、Amazon Simple Storage Service (Amazon S3) バケットをコンテンツオリジンとして使用しています。

コードは、ディストリビューションの作成後に CloudFrontWaiter を作成し、ディストリビューションがデプロイされるまで待ってからディストリビューションを返します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateDistribution {

        private static final Logger logger =
 LoggerFactory.getLogger(CreateDistribution.class);

        public static Distribution createDistribution(CloudFrontClient
 cloudFrontClient, S3Client s3Client,
                        final String bucketName, final String keyGroupId, final
 String originAccessControlId) {

                final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
                                .get("x-amz-bucket-region").get(0);
                final String originDomain = bucketName + ".s3." + region +
 ".amazonaws.com";
                String originId = originDomain; // Use the originDomain value for
 the originId.

                // The service API requires some deprecated methods, such as
                // DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
                CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                                .distributionConfig(b1 -> b1
                                                .origins(b2 -> b2
                                                                .quantity(1)
                                                                .items(b3 -> b3

 .domainName(originDomain)

 .id(originId)
```

```
.s3OriginConfig(builder4 -> builder4

              .originAccessIdentity(

                                 ""))

.originAccessControlId(

              originAccessControlId)))
                                                  .defaultCacheBehavior(b2 -> b2

.viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)
                                                                        .minTTL(200L)

.forwardedValues(b5 -> b5

.cookies(cp -> cp

              .forward(ItemSelection.NONE))

.queryString(true))

.trustedKeyGroups(b3 -> b3

.quantity(1)

.items(keyGroupId)

.enabled(true))

.allowedMethods(b4 -> b4

.quantity(2)

.items(Method.HEAD, Method.GET)

.cachedMethods(b5 -> b5

              .quantity(2)

              .items(Method.HEAD,
```

```
                                Method.GET))))
                                                .cacheBehaviors(b -> b
                                                                .quantity(1)
                                                                .items(b2 -> b2

.pathPattern("/index.html")

.viewerProtocolPolicy(

              ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3

              .quantity(1)

              .items(keyGroupId)

              .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4

              .cookies(cp -> cp

                              .forward(ItemSelection.NONE))

              .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)

              .items(Method.HEAD,

                              Method.GET)

              .cachedMethods(b6 -> b6

                              .quantity(2)

                              .items(Method.HEAD,
```

```
                                                Method.GET)))))
                                                 .enabled(true)
                                                 .comment("Distribution built with
 java")

 .callerReference(Instant.now().toString())));

                final Distribution distribution =
 createDistResponse.distribution();
                logger.info("Distribution created. DomainName: [{}]  Id: [{}]",
 distribution.domainName(),
                                distribution.id());
                logger.info("Waiting for distribution to be deployed ...");
                try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                                .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                                                .matched();
                        responseOrException.response()
                                                .orElseThrow(() -> new
 RuntimeException("Distribution not created"));
                        logger.info("Distribution deployed. DomainName: [{}]  Id:
 [{}]", distribution.domainName(),
                                                distribution.id());
                }
                return distribution;
        }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「CreateDistribution」を参照してください。

PowerShell

Tools for PowerShell

例 1: ロギングとキャッシュを使用して設定した、基本的な CloudFront ディストリビューションを作成します。

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "ps-cmdlet-sample.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
      -DistributionConfig_Enabled $true `
      -DistributionConfig_Comment "Test distribution" `
      -Origins_Item $origin `
      -Origins_Quantity 1 `
      -Logging_Enabled $true `
      -Logging_IncludeCookie $true `
      -Logging_Bucket ps-cmdlet-sample-logging.s3.amazonaws.com `
      -Logging_Prefix "help/" `
      -DistributionConfig_CallerReference Client1 `
      -DistributionConfig_DefaultRootObject index.html `
      -DefaultCacheBehavior_TargetOriginId $origin.Id `
      -ForwardedValues_QueryString $true `
      -Cookies_Forward all `
      -WhitelistedNames_Quantity 0 `
      -TrustedSigners_Enabled $false `
      -TrustedSigners_Quantity 0 `
      -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
      -DefaultCacheBehavior_MinTTL 1000 `
      -DistributionConfig_PriceClass "PriceClass_All" `
      -CacheBehaviors_Quantity 0 `
      -Aliases_Quantity 0
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet リファレンス」の「CreateDistribution」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CreateFunction** を使用する

次のコード例は、CreateFunction を使用する方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateFunction {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <functionName> <filePath>

                Where:
                    functionName - The name of the function to create.\s
                    filePath - The path to a file that contains the application
 logic for the function.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        String funArn = createNewFunction(cloudFrontClient, functionName,
filePath);
        System.out.println("The function ARN is " + funArn);
        cloudFrontClient.close();
    }

    public static String createNewFunction(CloudFrontClient cloudFrontClient,
String functionName, String filePath) {
        try {
            InputStream fileIs =
CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
            SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

            FunctionConfig config = FunctionConfig.builder()
                    .comment("Created by using the CloudFront Java API")
                    .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
                    .build();

            CreateFunctionRequest functionRequest =
CreateFunctionRequest.builder()
                    .name(functionName)
                    .functionCode(functionCode)
                    .functionConfig(config)
                    .build();

            CreateFunctionResponse response =
cloudFrontClient.createFunction(functionRequest);
            return response.functionSummary().functionMetadata().functionARN();

        } catch (CloudFrontException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
        return "";
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「CreateFunction」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CreateInvalidation** を使用する

以下のコード例は、CreateInvalidation の使用方法を示しています。

CLI

AWS CLI

CloudFront ディストリビューションでキャッシュ削除を作成するには

次の create-invalidation 例では、指定した CloudFront ディストリビューションで指定したファイルのキャッシュ削除を作成します。

```
aws cloudfront create-invalidation \
    --distribution-id EDFDVBD6EXAMPLE \
    --paths "/example-path/example-file.jpg" "/example-path/example-file2.png"
```

出力:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EDFDVBD6EXAMPLE/invalidation/I1JLWSDAP8FU89",
    "Invalidation": {
        "Id": "I1JLWSDAP8FU89",
        "Status": "InProgress",
        "CreateTime": "2019-12-05T18:24:51.407Z",
        "InvalidationBatch": {
```

```
            "Paths": {
                "Quantity": 2,
                "Items": [
                    "/example-path/example-file2.png",
                    "/example-path/example-file.jpg"
                ]
            },
            "CallerReference": "cli-1575570291-670203"
        }
    }
}
```

前の例では、AWS CLI が自動的にランダム CallerReference を生成しました。独自の CallerReference を指定したり、キャッシュ削除パラメータをコマンドライン引数として渡さないようにしたりするには、JSON ファイルを使用できます。次の例では、inv-batch.json という名前の JSON ファイルでキャッシュ削除パラメータを指定して、2 つのファイルのキャッシュ削除を作成します。

```
aws cloudfront create-invalidation \
    --distribution-id EDFDVBD6EXAMPLE \
    --invalidation-batch file://inv-batch.json
```

inv-batch.json の内容:

```
{
    "Paths": {
        "Quantity": 2,
        "Items": [
            "/example-path/example-file.jpg",
            "/example-path/example-file2.png"
        ]
    },
    "CallerReference": "cli-example"
}
```

出力:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EDFDVBD6EXAMPLE/invalidation/I2J0I21PCUYOIK",
    "Invalidation": {
```

```
        "Id": "I2J0I21PCUYOIK",
        "Status": "InProgress",
        "CreateTime": "2019-12-05T18:40:49.413Z",
        "InvalidationBatch": {
            "Paths": {
                "Quantity": 2,
                "Items": [
                    "/example-path/example-file.jpg",
                    "/example-path/example-file2.png"
                ]
            },
            "CallerReference": "cli-example"
        }
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「CreateInvalidation」を参照してください。

PowerShell

Tools for PowerShell

例 1: この例では、ID が EXAMPLENSTXAXE であるディストリビューションで、新しいキャッシュ削除を作成します。CallerReference はユーザーが選択した固有の ID です。この例では、2019 年 5 月 15 日の午前 9 時を表すタイムスタンプを使用しています。$Paths 変数には、ユーザーがディストリビューションのキャッシュに含めたくない画像ファイルやメディアファイルへの 3 つのパスが格納されます。-Paths_Quantity パラメータ値は、-Paths_Item パラメータで指定したパスの総数です。

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -
Paths_Quantity 3
```

出力:

```
Invalidation                         Location

------------                         --------
```

```
Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet リファレンス」の「CreateInvalidation」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CreateKeyGroup** を使用する

次のコード例は、CreateKeyGroup を使用する方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

キーグループには、署名付き URL または Cookie の検証に使用されるパブリックキーが少なくとも 1 つ必要です。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

public class CreateKeyGroup {
    private static final Logger logger =
 LoggerFactory.getLogger(CreateKeyGroup.class);

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
 publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b ->
 b.keyGroupConfig(c -> c
```

```
                .items(publicKeyId)
                .name("JavaKeyGroup" + UUID.randomUUID()))
                .keyGroup().id();
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「CreateKeyGroup」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CreatePublicKey** を使用する

以下のコード例は、CreatePublicKey の使用方法を示しています。

CLI

AWS CLI

CloudFront パブリックキーを作成するには

次の例では、pub-key-config.json という名前の JSON ファイルでパラメータを指定して、CloudFront パブリックキーを作成します。このコマンドを使用するには、事前に PEM でエンコードされたパブリックキーが必要です。詳細については、「Amazon CloudFront 開発者ガイド」の「RSA キーペアを作成する」を参照してください。

```
aws cloudfront create-public-key \
    --public-key-config file://pub-key-config.json
```

pub-key-config.json ファイルは、以下を含む現在のフォルダ内にある JSON ドキュメントです。パブリックキーは PEM 形式でエンコードされていることに注意してください。

```
{
    "CallerReference": "cli-example",
    "Name": "ExampleKey",
```

```
    "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPMbCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1UOWcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAzO3ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95ylUQd9LfYTBzVZYG2tSesplcOkjM3\n2Uu
+oMWxQAw1NINnSLPinMVsutJy6ZqlV3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLKOX5Rgb/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nrwIDAQAB\n-----
END PUBLIC KEY-----\n",
    "Comment": "example public key"
}
```

出力:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/public-key/
KDFB19YGCR002",
    "ETag": "E2QWRUHEXAMPLE",
    "PublicKey": {
        "Id": "KDFB19YGCR002",
        "CreatedTime": "2019-12-05T18:51:43.781Z",
        "PublicKeyConfig": {
            "CallerReference": "cli-example",
            "Name": "ExampleKey",
            "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPMbCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1UOWcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAzO3ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95ylUQd9LfYTBzVZYG2tSesplcOkjM3\n2Uu
+oMWxQAw1NINnSLPinMVsutJy6ZqlV3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLKOX5Rgb/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nrwIDAQAB\n-----
END PUBLIC KEY-----\n",
            "Comment": "example public key"
        }
    }
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「CreatePublicKey」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

次のコード例では、パブリックキーを読み込んで Amazon CloudFront にアップロードします。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

public class CreatePublicKey {
    private static final Logger logger =
 LoggerFactory.getLogger(CreatePublicKey.class);

    public static String createPublicKey(CloudFrontClient cloudFrontClient,
 String publicKeyFileName) {
        try (InputStream is =
 CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
                    .createPublicKey(b -> b.publicKeyConfig(c -> c
                            .name("JavaCreatedPublicKey" + UUID.randomUUID())
                            .encodedKey(publicKeyString)
                            .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
```

```
            throw new RuntimeException(e);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「CreatePublicKey」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteDistribution** を使用する

以下のコード例は、DeleteDistribution の使用方法を示しています。

CLI

AWS CLI

CloudFront ディストリビューションを削除するには

次の例では、ID EDFDVBD6EXAMPLE の付いた CloudFront ディストリビューションを削除します。ディストリビューションを削除する前に、ディストリビューションを無効にする必要があります。ディストリビューションを無効にするには、update-distribution コマンドを使用します。詳細については、update-distribution の例を参照してください。

ディストリビューションを無効にすると、ディストリビューションを削除できます。ディストリビューションを削除するには、--if-match オプションを使用してディストリビューション のETag を指定する必要があります。ETag を取得するには、get-distribution コマンドまたは get-distribution-config コマンドを使用します。

```
aws cloudfront delete-distribution \
    --id EDFDVBD6EXAMPLE \
    --if-match E2QWRUHEXAMPLE
```

成功した場合は、コマンドの出力はありません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteDistribution」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

次のコード例では、配信を無効状態に更新し、ウェイターを使用して変更がデプロイされるのを待ってから、ディストリビューションを削除します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

public class DeleteDistribution {
        private static final Logger logger =
 LoggerFactory.getLogger(DeleteDistribution.class);

        public static void deleteDistribution(final CloudFrontClient
 cloudFrontClient, final String distributionId) {
                // First, disable the distribution by updating it.
                GetDistributionResponse response =
 cloudFrontClient.getDistribution(b -> b
                                .id(distributionId));
                String etag = response.eTag();
                DistributionConfig distConfig =
 response.distribution().distributionConfig();

                cloudFrontClient.updateDistribution(builder -> builder
                                .id(distributionId)
                                .distributionConfig(builder1 -> builder1

 .cacheBehaviors(distConfig.cacheBehaviors())
```

```
.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                                                .enabled(false)
                                                .origins(distConfig.origins())
                                                .comment(distConfig.comment())

.callerReference(distConfig.callerReference())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())

.priceClass(distConfig.priceClass())
                                                .aliases(distConfig.aliases())
                                                .logging(distConfig.logging())

.defaultRootObject(distConfig.defaultRootObject())

.customErrorResponses(distConfig.customErrorResponses())

.httpVersion(distConfig.httpVersion())

.isIPV6Enabled(distConfig.isIPV6Enabled())

.restrictions(distConfig.restrictions())

.viewerCertificate(distConfig.viewerCertificate())
                                                .webACLId(distConfig.webACLId())

.originGroups(distConfig.originGroups()))
                                .ifMatch(etag));

                logger.info("Distribution [{}] is DISABLED, waiting for
deployment before deleting ...",
                                distributionId);
                GetDistributionResponse distributionResponse;
                try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter
                                        .waitUntilDistributionDeployed(builder ->
builder.id(distributionId)).matched();
                        distributionResponse = responseOrException.response()
                                        .orElseThrow(() -> new
RuntimeException("Could not disable distribution"));
                }
```

```
                DeleteDistributionResponse deleteDistributionResponse =
 cloudFrontClient
                                .deleteDistribution(builder -> builder
                                                .id(distributionId)

 .ifMatch(distributionResponse.eTag()));
                if (deleteDistributionResponse.sdkHttpResponse().isSuccessful())
 {
                        logger.info("Distribution [{}] DELETED", distributionId);
                }
        }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「DeleteDistribution」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetCloudFrontOriginAccessIdentity** を使用する

以下のコード例は、GetCloudFrontOriginAccessIdentity の使用方法を示しています。

CLI

AWS CLI

CloudFront オリジンアクセスアイデンティティを取得するには

次の例では、ID が E74FTE3AEXAMPLE である CloudFront オリジンアクセスアイデンティティ (OAI) を、ETag および関連する S3 正規 ID を含めて取得します。OAI ID は、create-cloud-front-origin-access-identity コマンドと list-cloud-front-origin-access-identities コマンドの出力で返されます。

```
aws cloudfront get-cloud-front-origin-access-identity --id E74FTE3AEXAMPLE
```

出力:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "CloudFrontOriginAccessIdentity": {
        "Id": "E74FTE3AEXAMPLE",
        "S3CanonicalUserId":
 "cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
        "CloudFrontOriginAccessIdentityConfig": {
            "CallerReference": "cli-example",
            "Comment": "Example OAI"
        }
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetCloudFrontOriginAccessIdentity」を参照してください。

PowerShell

Tools for PowerShell

例 1: この例では、-Id パラメータで指定した、特定の Amazon CloudFront オリジンアクセスアイデンティティを返します。-Id パラメータは必須ではありませんが、指定しないと、結果は返されません。

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

出力:

```
     CloudFrontOriginAccessIdentityConfig    Id
    S3CanonicalUserId
     ------------------------------------    --
    -----------------
     Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
    4b6e...
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet リファレンス」の「GetCloudFrontOriginAccessIdentity」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で `GetCloudFrontOriginAccessIdentityConfig` を使用する

以下のコード例は、`GetCloudFrontOriginAccessIdentityConfig` の使用方法を示しています。

CLI

AWS CLI

CloudFront オリジンアクセスアイデンティティ設定を取得するには

次の例では、ID が `E74FTE3AEXAMPLE` である CloudFront オリジンアクセスアイデンティティ (OAI) に関するメタデータ (`ETag` を含む) を取得します。OAI ID は、create-cloud-front-origin-access-identity コマンドと list-cloud-front-origin-access-identities コマンドの出力で返されます。

```
aws cloudfront get-cloud-front-origin-access-identity-config --id E74FTE3AEXAMPLE
```

出力:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "CloudFrontOriginAccessIdentityConfig": {
        "CallerReference": "cli-example",
        "Comment": "Example OAI"
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetCloudFrontOriginAccessIdentityConfig」を参照してください。

PowerShell

Tools for PowerShell

例 1: この例では、-Id パラメータで指定した、単一の Amazon CloudFront オリジンアクセスアイデンティティに関する設定情報を返します。-Id パラメータを指定しないと、エラーが発生します。

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

出力:

```
     CallerReference                                                Comment
     ---------------                                                -------
     mycallerreference: 2/1/2011 1:16:32 PM                         Caller
reference: 2/1/2011 1:16:32 PM
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet リファレンス」の「GetCloudFrontOriginAccessIdentityConfig」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetDistribution** を使用する

以下のコード例は、GetDistribution の使用方法を示しています。

CLI

AWS CLI

CloudFront ディストリビューションを取得するには

次の例では、ID が EDFDVBD6EXAMPLE である CloudFront ディストリビューション (ETag を含む) を取得します。ディストリビューション ID は create-distribution コマンドと list-distributions コマンドで返されます。

```
aws cloudfront get-distribution --id EDFDVBD6EXAMPLE
```

出力:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "Distribution": {
        "Id": "EDFDVBD6EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
        "Status": "Deployed",
        "LastModifiedTime": "2019-12-04T23:35:41.433Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
                        "DomainName": "awsexamplebucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-
example",
                "ForwardedValues": {
```

```
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
```

```
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetDistribution」を参照してください。

PowerShell

Tools for PowerShell

例 1: 特定のディストリビューションに関する情報を取得します。

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet リファレンス」の「GetDistribution」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetDistributionConfig** を使用する

以下のコード例は、GetDistributionConfig の使用方法を示しています。

CLI

AWS CLI

CloudFront ディストリビューション設定を取得するには

次の例では、ID EDFDVBD6EXAMPLE の CloudFront ディストリビューションに関するメタデータ (ETag を含む) を取得します。ディストリビューション ID は create-distribution コマンドと list-distributions コマンドで返されます。

```
aws cloudfront get-distribution-config --id EDFDVBD6EXAMPLE
```

出力:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "DistributionConfig": {
        "CallerReference": "cli-example",
        "Aliases": {
            "Quantity": 0
        },
        "DefaultRootObject": "index.html",
        "Origins": {
            "Quantity": 1,
            "Items": [
                {
                    "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
                    "DomainName": "awsexamplebucket.s3.amazonaws.com",
                    "OriginPath": "",
                    "CustomHeaders": {
                        "Quantity": 0
                    },
                    "S3OriginConfig": {
                        "OriginAccessIdentity": ""
                    }
```

```
                }
            ]
        },
        "OriginGroups": {
            "Quantity": 0
        },
        "DefaultCacheBehavior": {
            "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
            "ForwardedValues": {
                "QueryString": false,
                "Cookies": {
                    "Forward": "none"
                },
                "Headers": {
                    "Quantity": 0
                },
                "QueryStringCacheKeys": {
                    "Quantity": 0
                }
            },
            "TrustedSigners": {
                "Enabled": false,
                "Quantity": 0
            },
            "ViewerProtocolPolicy": "allow-all",
            "MinTTL": 0,
            "AllowedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ],
                "CachedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ]
                }
            },
            "SmoothStreaming": false,
            "DefaultTTL": 86400,
            "MaxTTL": 31536000,
            "Compress": false,
```

```
            "LambdaFunctionAssociations": {
                "Quantity": 0
            },
            "FieldLevelEncryptionId": ""
        },
        "CacheBehaviors": {
            "Quantity": 0
        },
        "CustomErrorResponses": {
            "Quantity": 0
        },
        "Comment": "",
        "Logging": {
            "Enabled": false,
            "IncludeCookies": false,
            "Bucket": "",
            "Prefix": ""
        },
        "PriceClass": "PriceClass_All",
        "Enabled": true,
        "ViewerCertificate": {
            "CloudFrontDefaultCertificate": true,
            "MinimumProtocolVersion": "TLSv1",
            "CertificateSource": "cloudfront"
        },
        "Restrictions": {
            "GeoRestriction": {
                "RestrictionType": "none",
                "Quantity": 0
            }
        },
        "WebACLId": "",
        "HttpVersion": "http2",
        "IsIPV6Enabled": true
    }
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「GetDistributionConfig」を参照してください。

PowerShell

Tools for PowerShell

例 1: 特定のディストリビューションの設定を取得します。

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet リファレンス」の「GetDistributionConfig」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
 self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
```

```
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「GetDistributionConfig」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ListCloudFrontOriginAccessIdentities** を使用する

以下のコード例は、`ListCloudFrontOriginAccessIdentities` の使用方法を示しています。

CLI

AWS CLI

CloudFront オリジンアクセスアイデンティティを一覧表示するには

次の例では、AWS アカウント内の CloudFront オリジンアクセスアイデンティティ (OAI) の一覧を取得します。

```
aws cloudfront list-cloud-front-origin-access-identities
```

出力:

```
{
    "CloudFrontOriginAccessIdentityList": {
        "Items": [
            {
                "Id": "E74FTE3AEXAMPLE",
                "S3CanonicalUserId":
 "cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
                "Comment": "Example OAI"
            },
            {
                "Id": "EH1HDMBEXAMPLE",
                "S3CanonicalUserId":
 "1489f6f2e6faacaae7ff64c4c3e6956c24f78788abfc1718c3527c263bf7a17EXAMPLE",
                "Comment": "Test OAI"
            },
            {
                "Id": "E2X2C9TEXAMPLE",
                "S3CanonicalUserId":
 "cbfeebb915a64749f9be546a45b3fcfd3a31c779673c13c4dd460911ae402c2EXAMPLE",
                "Comment": "Example OAI #2"
            }
        ]
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「ListCloudFrontOriginAccessIdentities」を参照してください。

PowerShell

Tools for PowerShell

例 1: この例では、Amazon CloudFront オリジンアクセスアイデンティティのリストを返します。-MaxItem パラメータは値として 2 を指定しているため、結果には 2 つのアイデンティティが含まれます。

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

出力:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet リファレンス」の「ListCloudFrontOriginAccessIdentities」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ListDistributions** を使用する

以下のコード例は、ListDistributions の使用方法を示しています。

CLI

AWS CLI

CloudFront ディストリビューションを一覧表示するには

次の例は、AWS アカウント内の CloudFront ディストリビューションの一覧を取得します。

```
aws cloudfront list-distributions
```

出力:

```
{
    "DistributionList": {
        "Items": [
            {
                "Id": "EMLARXS9EXAMPLE",
                "ARN": "arn:aws:cloudfront::123456789012:distribution/
EMLARXS9EXAMPLE",
                "Status": "InProgress",
                "LastModifiedTime": "2019-11-22T00:55:15.705Z",
                "InProgressInvalidationBatches": 0,
                "DomainName": "d111111abcdef8.cloudfront.net",
```

```
                "ActiveTrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "DistributionConfig": {
                    "CallerReference": "cli-example",
                    "Aliases": {
                        "Quantity": 0
                    },
                    "DefaultRootObject": "index.html",
                    "Origins": {
                        "Quantity": 1,
                        "Items": [
                            {
                                "Id": "awsexamplebucket.s3.amazonaws.com-cli-
example",
                                "DomainName":
 "awsexamplebucket.s3.amazonaws.com",
                                "OriginPath": "",
                                "CustomHeaders": {
                                    "Quantity": 0
                                },
                                "S3OriginConfig": {
                                    "OriginAccessIdentity": ""
                                }
                            }
                        ]
                    },
                    "OriginGroups": {
                        "Quantity": 0
                    },
                    "DefaultCacheBehavior": {
                        "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-
example",
                        "ForwardedValues": {
                            "QueryString": false,
                            "Cookies": {
                                "Forward": "none"
                            },
                            "Headers": {
                                "Quantity": 0
                            },
                            "QueryStringCacheKeys": {
                                "Quantity": 0
```

```
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
```

```
                    "PriceClass": "PriceClass_All",
                    "Enabled": true,
                    "ViewerCertificate": {
                        "CloudFrontDefaultCertificate": true,
                        "MinimumProtocolVersion": "TLSv1",
                        "CertificateSource": "cloudfront"
                    },
                    "Restrictions": {
                        "GeoRestriction": {
                            "RestrictionType": "none",
                            "Quantity": 0
                        }
                    },
                    "WebACLId": "",
                    "HttpVersion": "http2",
                    "IsIPV6Enabled": true
                }
            },
            {
                "Id": "EDFDVBD6EXAMPLE",
                "ARN": "arn:aws:cloudfront::123456789012:distribution/
EDFDVBD6EXAMPLE",
                "Status": "InProgress",
                "LastModifiedTime": "2019-12-04T23:35:41.433Z",
                "InProgressInvalidationBatches": 0,
                "DomainName": "d930174dauwrn8.cloudfront.net",
                "ActiveTrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "DistributionConfig": {
                    "CallerReference": "cli-example",
                    "Aliases": {
                        "Quantity": 0
                    },
                    "DefaultRootObject": "index.html",
                    "Origins": {
                        "Quantity": 1,
                        "Items": [
                            {
                                "Id": "awsexamplebucket1.s3.amazonaws.com-cli-
example",
                                "DomainName":
 "awsexamplebucket1.s3.amazonaws.com",
```

```
                                "OriginPath": "",
                                "CustomHeaders": {
                                    "Quantity": 0
                                },
                                "S3OriginConfig": {
                                    "OriginAccessIdentity": ""
                                }
                            }
                        ]
                    },
                    "OriginGroups": {
                        "Quantity": 0
                    },
                    "DefaultCacheBehavior": {
                        "TargetOriginId": "awsexamplebucket1.s3.amazonaws.com-
cli-example",
                        "ForwardedValues": {
                            "QueryString": false,
                            "Cookies": {
                                "Forward": "none"
                            },
                            "Headers": {
                                "Quantity": 0
                            },
                            "QueryStringCacheKeys": {
                                "Quantity": 0
                            }
                        },
                        "TrustedSigners": {
                            "Enabled": false,
                            "Quantity": 0
                        },
                        "ViewerProtocolPolicy": "allow-all",
                        "MinTTL": 0,
                        "AllowedMethods": {
                            "Quantity": 2,
                            "Items": [
                                "HEAD",
                                "GET"
                            ],
                            "CachedMethods": {
                                "Quantity": 2,
                                "Items": [
                                    "HEAD",
```

```
                                "GET"
                            ]
                        }
                    },
                    "SmoothStreaming": false,
                    "DefaultTTL": 86400,
                    "MaxTTL": 31536000,
                    "Compress": false,
                    "LambdaFunctionAssociations": {
                        "Quantity": 0
                    },
                    "FieldLevelEncryptionId": ""
                },
                "CacheBehaviors": {
                    "Quantity": 0
                },
                "CustomErrorResponses": {
                    "Quantity": 0
                },
                "Comment": "",
                "Logging": {
                    "Enabled": false,
                    "IncludeCookies": false,
                    "Bucket": "",
                    "Prefix": ""
                },
                "PriceClass": "PriceClass_All",
                "Enabled": true,
                "ViewerCertificate": {
                    "CloudFrontDefaultCertificate": true,
                    "MinimumProtocolVersion": "TLSv1",
                    "CertificateSource": "cloudfront"
                },
                "Restrictions": {
                    "GeoRestriction": {
                        "RestrictionType": "none",
                        "Quantity": 0
                    }
                },
                "WebACLId": "",
                "HttpVersion": "http2",
                "IsIPV6Enabled": true
            }
        },
```

```
            {
                "Id": "E1X5IZQEXAMPLE",
                "ARN": "arn:aws:cloudfront::123456789012:distribution/
E1X5IZQEXAMPLE",
                "Status": "Deployed",
                "LastModifiedTime": "2019-11-06T21:31:48.864Z",
                "DomainName": "d2e04y12345678.cloudfront.net",
                "Aliases": {
                    "Quantity": 0
                },
                "Origins": {
                    "Quantity": 1,
                    "Items": [
                        {
                            "Id": "awsexamplebucket2",
                            "DomainName": "awsexamplebucket2.s3.us-
west-2.amazonaws.com",
                            "OriginPath": "",
                            "CustomHeaders": {
                                "Quantity": 0
                            },
                            "S3OriginConfig": {
                                "OriginAccessIdentity": ""
                            }
                        }
                    ]
                },
                "OriginGroups": {
                    "Quantity": 0
                },
                "DefaultCacheBehavior": {
                    "TargetOriginId": "awsexamplebucket2",
                    "ForwardedValues": {
                        "QueryString": false,
                        "Cookies": {
                            "Forward": "none"
                        },
                        "Headers": {
                            "Quantity": 0
                        },
                        "QueryStringCacheKeys": {
                            "Quantity": 0
                        }
                    },
```

```
                    "TrustedSigners": {
                        "Enabled": false,
                        "Quantity": 0
                    },
                    "ViewerProtocolPolicy": "allow-all",
                    "MinTTL": 0,
                    "AllowedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ],
                        "CachedMethods": {
                            "Quantity": 2,
                            "Items": [
                                "HEAD",
                                "GET"
                            ]
                        }
                    },
                    "SmoothStreaming": false,
                    "DefaultTTL": 86400,
                    "MaxTTL": 31536000,
                    "Compress": false,
                    "LambdaFunctionAssociations": {
                        "Quantity": 0
                    },
                    "FieldLevelEncryptionId": ""
                },
                "CacheBehaviors": {
                    "Quantity": 0
                },
                "CustomErrorResponses": {
                    "Quantity": 0
                },
                "Comment": "",
                "PriceClass": "PriceClass_All",
                "Enabled": true,
                "ViewerCertificate": {
                    "CloudFrontDefaultCertificate": true,
                    "MinimumProtocolVersion": "TLSv1",
                    "CertificateSource": "cloudfront"
                },
                "Restrictions": {
```

```
                    "GeoRestriction": {
                        "RestrictionType": "none",
                        "Quantity": 0
                    }
                },
                "WebACLId": "",
                "HttpVersion": "HTTP1_1",
                "IsIPV6Enabled": true
            }
        ]
    }
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「ListDistributions」を参照してください。

PowerShell

Tools for PowerShell

例 1: ディストリビューションを返します。

```
Get-CFDistributionList
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet リファレンス」の「ListDistributions」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""
```

```
    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def list_distributions(self):
        print("CloudFront distributions:\n")
        distributions = self.cloudfront_client.list_distributions()
        if distributions["DistributionList"]["Quantity"] > 0:
            for distribution in distributions["DistributionList"]["Items"]:
                print(f"Domain: {distribution['DomainName']}")
                print(f"Distribution Id: {distribution['Id']}")
                print(
                    f"Certificate Source: "
                    f"{distribution['ViewerCertificate']['CertificateSource']}"
                )
                if distribution["ViewerCertificate"]["CertificateSource"] ==
 "acm":
                    print(
                        f"Certificate: {distribution['ViewerCertificate']
['Certificate']}"
                    )
                print("")
        else:
            print("No CloudFront distributions detected.")
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「ListDistributions」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **UpdateDistribution** を使用する

以下のコード例は、UpdateDistribution の使用方法を示しています。

CLI

AWS CLI

CloudFront ディストリビューションのデフォルトルートオブジェクトを更新するには

次の例では、ID EDFDVBD6EXAMPLE を持つ CloudFront ディストリビューションのデフォルトルートオブジェクトを index.html に更新します。

```
aws cloudfront update-distribution --id EDFDVBD6EXAMPLE \
    --default-root-object index.html
```

出力:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "Distribution": {
        "Id": "EDFDVBD6EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-12-06T18:55:39.870Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "6b10378d-49be-4c4b-a642-419ccaf8f3b5",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "example-website",
                        "DomainName": "www.example.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
```

```
                    "CustomOriginConfig": {
                        "HTTPPort": 80,
                        "HTTPSPort": 443,
                        "OriginProtocolPolicy": "match-viewer",
                        "OriginSslProtocols": {
                            "Quantity": 2,
                            "Items": [
                                "SSLv3",
                                "TLSv1"
                            ]
                        },
                        "OriginReadTimeout": 30,
                        "OriginKeepaliveTimeout": 5
                    }
                }
            ]
        },
        "OriginGroups": {
            "Quantity": 0
        },
        "DefaultCacheBehavior": {
            "TargetOriginId": "example-website",
            "ForwardedValues": {
                "QueryString": false,
                "Cookies": {
                    "Forward": "none"
                },
                "Headers": {
                    "Quantity": 1,
                    "Items": [
                        "*"
                    ]
                },
                "QueryStringCacheKeys": {
                    "Quantity": 0
                }
            },
            "TrustedSigners": {
                "Enabled": false,
                "Quantity": 0
            },
            "ViewerProtocolPolicy": "allow-all",
            "MinTTL": 0,
            "AllowedMethods": {
```

```
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
```

```
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http1.1",
            "IsIPV6Enabled": true
        }
    }
}
```

CloudFront ディストリビューションを更新するには

次の例では、dist-config-disable.json という名前の JSON ファイルでディストリビューション設定を指定して、ID EMLARXS9EXAMPLE を持つ CloudFront ディストリビューションを無効にします。ディストリビューションを更新するには、--if-match オプションを使用してディストリビューションの ETag を指定する必要があります。ETag を取得するには、get-distribution コマンドまたは get-distribution-config コマンドを使用します。

次の例を使用してディストリビューションを無効にした後は、delete-distribution コマンドを使用してディストリビューションを削除できます。

```
aws cloudfront update-distribution \
    --id EMLARXS9EXAMPLE \
    --if-match E2QWRUHEXAMPLE \
    --distribution-config file://dist-config-disable.json
```

dist-config-disable.json ファイルは、以下を含む現在のフォルダ内にある JSON ドキュメントです。Enabled フィールドが false に設定されていることに注意してください。

```
{
    "CallerReference": "cli-1574382155-496510",
    "Aliases": {
        "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "awsexamplebucket.s3.amazonaws.com-1574382155-273939",
```

```
                "DomainName": "awsexamplebucket.s3.amazonaws.com",
                "OriginPath": "",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "OriginGroups": {
        "Quantity": 0
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-1574382155-273939",
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
            },
            "Headers": {
                "Quantity": 0
            },
            "QueryStringCacheKeys": {
                "Quantity": 0
            }
        },
        "TrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ViewerProtocolPolicy": "allow-all",
        "MinTTL": 0,
        "AllowedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
```

```
                    "GET"
                ]
            }
        },
        "SmoothStreaming": false,
        "DefaultTTL": 86400,
        "MaxTTL": 31536000,
        "Compress": false,
        "LambdaFunctionAssociations": {
            "Quantity": 0
        },
        "FieldLevelEncryptionId": ""
    },
    "CacheBehaviors": {
        "Quantity": 0
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Comment": "",
    "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": false,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
}
```

出力:


```
{
    "ETag": "E9LHASXEXAMPLE",
    "Distribution": {
        "Id": "EMLARXS9EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-12-06T18:32:35.553Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-1574382155-496510",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id":
 "awsexamplebucket.s3.amazonaws.com-1574382155-273939",
                        "DomainName": "awsexamplebucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId":
 "awsexamplebucket.s3.amazonaws.com-1574382155-273939",
```

```
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
```

```
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": false,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「UpdateDistribution」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import
 software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ModifyDistribution {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <id>\s

                Where:
                    id - the id value of the distribution.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String id = args[0];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        modDistribution(cloudFrontClient, id);
        cloudFrontClient.close();
    }
```

```
   public static void modDistribution(CloudFrontClient cloudFrontClient, String
idVal) {
       try {
           // Get the Distribution to modify.
           GetDistributionRequest disRequest = GetDistributionRequest.builder()
                   .id(idVal)
                   .build();

           GetDistributionResponse response =
cloudFrontClient.getDistribution(disRequest);
           Distribution disObject = response.distribution();
           DistributionConfig config = disObject.distributionConfig();

           // Create a new DistributionConfig object and add new values to
comment and
           // aliases
           DistributionConfig config1 = DistributionConfig.builder()
                   .aliases(config.aliases()) // You can pass in new values here
                   .comment("New Comment")
                   .cacheBehaviors(config.cacheBehaviors())
                   .priceClass(config.priceClass())
                   .defaultCacheBehavior(config.defaultCacheBehavior())
                   .enabled(config.enabled())
                   .callerReference(config.callerReference())
                   .logging(config.logging())
                   .originGroups(config.originGroups())
                   .origins(config.origins())
                   .restrictions(config.restrictions())
                   .defaultRootObject(config.defaultRootObject())
                   .webACLId(config.webACLId())
                   .httpVersion(config.httpVersion())
                   .viewerCertificate(config.viewerCertificate())
                   .customErrorResponses(config.customErrorResponses())
                   .build();

           UpdateDistributionRequest updateDistributionRequest =
UpdateDistributionRequest.builder()
                   .distributionConfig(config1)
                   .id(disObject.id())
                   .ifMatch(response.eTag())
                   .build();

           cloudFrontClient.updateDistribution(updateDistributionRequest);
```

```
        } catch (CloudFrontException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「UpdateDistribution」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
 self.cloudfront_client.get_distribution_config(
            Id=distribution_id
```

```
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「UpdateDistribution」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

SDK AWS を使用する CloudFront のシナリオ

以下のコード例は、AWS SDK を使用して CloudFront で一般的なシナリオを実装する方法を示しています。これらのシナリオは、CloudFront 内で複数の関数を呼び出して特定のタスクを実行する方法を示しています。それぞれのシナリオには、GitHub へのリンクがあり、コードを設定および実行する方法についての説明が記載されています。

例

- SDKAWS を使用して CloudFront 署名リソースを削除する
- AWS SDK を使用して署名付き URL と Cookie を作成する

SDKAWS を使用して CloudFront 署名リソースを削除する

次のコード例は、Amazon Simple Storage Service (Amazon S3) バケット内の制限付きコンテンツへのアクセス権を取得するために使用されるリソースを削除する方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

public class DeleteSigningResources {
    private static final Logger logger =
 LoggerFactory.getLogger(DeleteSigningResources.class);

    public static void deleteOriginAccessControl(final CloudFrontClient
 cloudFrontClient,
            final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
                .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
 cloudFrontClient.deleteOriginAccessControl(builder -> builder
                .id(originAccessControlId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
```

```
            logger.info("Successfully deleted Origin Access Control [{}]",
 originAccessControlId);
        }
    }

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient,
 final String keyGroupId) {

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
 b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
 cloudFrontClient.deleteKeyGroup(builder -> builder
                .id(keyGroupId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
 final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
 b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
 cloudFrontClient.deletePublicKey(builder -> builder
                .id(publicKeyId)
                .ifMatch(getResponse.eTag()));

        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - DeleteKeyGroup
  - DeleteOriginAccessControl
  - DeletePublicKey

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して署名付き URL と Cookie を作成する

次のコード例は、制限付きリソースへのアクセスを許可する署名付き URL と Cookie を作成する方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CannedSignerRequest クラスを使用すると、既定ポリシーを使用して URL または Cookies に署名できます。

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
```

```
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expirationDate)
                .build();
    }
}
```

CustomSignerRequest クラスを使用すると、カスタムポリシーを使用して URL や Cookie に署名できます。activeDate および ipRange はオプションのメソッドです。

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
```

```
                .expirationDate(expireDate)
                .activeDate(activeDate) // Optional.
                // .ipRange("192.168.0.1/24") // Optional.
                .build();
    }
}
```

次の例は、CloudFrontUtilities クラスを使用して署名付き Cookies と URL を生成する方法を示しています。このコード例を GitHub を表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
 LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
 CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
 cannedSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
 customSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }
```

```
    public static CookiesForCannedPolicy
 getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
                .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
 cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCannedPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }

    public static CookiesForCustomPolicy
 getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
        CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
                .getCookiesForCustomPolicy(customSignerRequest);
        logger.info("Cookie POLICY header [{}]",
 cookiesForCustomPolicy.policyHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCustomPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCustomPolicy.signatureHeaderValue());
        return cookiesForCustomPolicy;
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「CloudFrontUtilities」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「AWS SDK での CloudFront の使用」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

ドキュメント履歴

次の表は、CloudFront ドキュメントに行われた重要な変更点をまとめたものです。更新の通知を受け取る場合は、RSS フィードにサブスクライブできます。

変更	説明	日付
新しいマネージドキャッシュポリシーの追加	新しいマネージドキャッシュポリシー `UseOriginCacheControlHeaders` と `UseOriginCacheControlHeaders-QueryStrings` を追加しました。	2024 年 5 月 20 日
オリジンアクセスコントロールサポートの追加	AWS Elemental MediaPackage V2 と AWS Lambda 関数 URL のオリジンアクセスコントロール (OAC) を作成できるようになりました。	2024 年 4 月 11 日
CMCD のリアルタイムログフィールド	リアルタイムログ用に 18 個の一般的なメディアクライアントデータ (CMCD) フィールドを追加しました。	2024 年 4 月 9 日
基本的な CloudFront ディストリビューションの開始方法	オリジンアクセスコントロール (OAC) で Amazon S3 オリジンを使用する基本的なディストリビューションのチュートリアルを更新しました。	2024 年 3 月 18 日
AWS SDK を使用した CloudFront のコード例	AWS Software Development Kit (SDK) で CloudFront を使用する方法を示すコード例を追加しました。例は、個々のサービス関数を呼び出す方法を示すコード抜粋と、同じサ	2024 年 2 月 16 日

	ービス内で複数の関数を呼び出して特定のタスクを達成する方法を示す例に分けられています。	
AWS マネージドポリシーの更新	`CloudFrontReadOnlyAccess` と `CloudFrontFullAccess` IAM ポリシーが `KeyValueStore` オペレーションをサポートするようになりました。	2023 年 12 月 19 日
JavaScript ランタイム 2.0	CloudFront Functions の JavaScript ランタイム 2.0 の機能を追加しました。	2023 年 11 月 21 日
CloudFront KeyValueStore	Amazon CloudFront は CloudFront KeyValueStore をサポートするようになりました。この機能は、CloudFront Functions 内からの読み取りアクセスを可能にする、安全でグローバルな、低レイテンシーのキー値データストアです。これにより、CloudFront エッジロケーションで高度にカスタマイズが可能なロジックを利用できるようになりました。	2023 年 11 月 21 日
Lambda@Edge がより新しいランタイムバージョンをサポート	Lambda@Edge は、Lambda 関数を Node.js 20 ランタイムでサポートするようになりました。	2023 年 11 月 15 日

セキュリティダッシュボード	ディストリビューションを作成すると、CloudFront はセキュリティダッシュボードを作成します。AWS WAF を有効にして、地理的制限を管理し、リクエスト、ボット、ログの概要データを表示します。	2023 年 11 月 8 日
関数内のクエリ文字列のソート	CloudFront は CloudFront Functions を使用したクエリ文字列のソートをサポートするようになりました。	2023 年 10 月 3 日
AWS WAF のセキュリティに関する推奨事項	Amazon CloudFront は、CloudFront コンソールに AWS WAF のセキュリティに関する推奨事項を表示するようになりました。	2023 年 9 月 26 日
古い (期限切れの) キャッシュコンテンツ提供のサポート	CloudFront は `Stale-While-Revalidate` および `Stale-If-Error` キャッシュ制御ディレクティブをサポートしています。	2023 年 5 月 15 日
ワンクリックで AWS WAF 保護を有効にする	CloudFront ディストリビューションに AWS WAF セキュリティ保護を追加するための効率的な方法です。	2023 年 5 月 10 日
標準ログに使用される新しい S3 バケットの ACL を有効にする	新しい S3 バケットのデフォルトの ACL 設定に対処するためのメモとリンクを追加しました。	2023 年 4 月 11 日

Amazon S3 Object Lambda を使用してオリジンを作成する	Amazon S3 Object Lambda アクセスポイントエイリアスをディストリビューションのオリジンとして使用できます。	2023 年 3 月 31 日
CloudFront Functions を使用して HTTP のステータスと本文をカスタマイズする	CloudFront Functions を使用して、ビューワーのレスポンスステータスコードを更新したり、レスポンス本文を置き換えたり削除したりできます。	2023 年 3 月 29 日
ポートの CORS ヘッダーのワイルドカードオプションを追加	CORS アクセスコントロールヘッダーで、ポートのワイルドカード設定を含めることができるようになりました。	2023 年 3 月 20 日
AWS Security Hub ユーザーガイドの新しいリンクを追加	言語を更新し、「AWS Security Hub ユーザーガイド」の「再編成された Amazon CloudFront コントロール」へのリンクを追加しました。	2023 年 3 月 9 日
CloudFront がオリジンリクエストポリシーでブロックリスト (「以外すべて」) をサポートするようになりました	オリジンリクエストポリシーのブロックリストを使用して、CloudFront がオリジンに送信するリクエストに、指定されたものを除くすべてのクエリ文字列、HTTP ヘッダー、または Cookie を含めます。	2023 年 2 月 22 日

CloudFront は、ホストヘッダーを除くすべてのビューワーヘッダーを転送する新しいマネージドオリジンリクエストポリシーを追加	CloudFront の新しいマネージドオリジンリクエストポリシーを使用して、CloudFront がオリジンに送信するリクエストに、Host ヘッダーを除くビューワーリクエストのすべてのヘッダーを含めます。	2023 年 2 月 22 日
Lambda@Edge に関する制限の更新	Lambda@Edge は、[Auto] (自動) に設定された Lambda ランタイム管理設定をサポートします。	2023 年 2 月 16 日
CloudFront の IAM ガイダンスの更新	IAM ベストプラクティスに沿ってガイドを更新しました。詳細については、「IAM のセキュリティのベストプラクティス」を参照してください。	2023 年 2 月 15 日
オリジンアクセスコントロールによるセキュリティ強化	指定した CloudFront ディストリビューションのみへのアクセスを許可することで、MediaStore オリジンを保護できるようになりました。	2023 年 2 月 9 日
ビューワーのヘッダー構造を決定するための新しいヘッダー	ビューワーを特定しやすくするために、ビューワーが送信するヘッダーに基づいてヘッダーの順序とヘッダー数を追加できるようになりました。	2023 年 1 月 13 日
Lambda@Edge がより新しいランタイムバージョンをサポート	Lambda@Edge は、Lambda 関数を Node.js 18 ランタイムでサポートするようになりました。	2023 年 1 月 12 日

レスポンスヘッダーポリシーを使用してレスポンスヘッダーを削除する	CloudFront レスポンスヘッダーポリシーを使用して、CloudFront がオリジンからのレスポンスで受け取ったヘッダーを削除できるようになりました。指定したヘッダーは、CloudFront からビューワーに送信するレスポンスに含まれなくなります。	2023 年 1 月 3 日
設定の変更を安全にテストするための継続的デプロイ	本番トラフィックのサブセットでテストすることで、CDN 設定に変更をデプロイできるようになりました。	2022 年 11 月 18 日
CloudFront-Viewer-JA3-Fingerprint ヘッダーのリリース	JA3 フィンガープリントを使用して、リクエストが既知のクライアントからのものかどうかを判断できるようになりました。	2022 年 11 月 16 日
CORS ヘッダーのワイルドカードオプションを追加	一部の CORS アクセスコントロールヘッダーで、さまざまなワイルドカード設定を使用できるようになりました。	2022 年 11 月 11 日
CloudFront ディストリビューション用の追加のメトリクス	CloudFront API と AWS CloudFormation での `MonitoringSubscription` のサポート。	2022 年 10 月 3 日
オリジンアクセスコントロールによるセキュリティ強化	指定した CloudFront ディストリビューションのみへのアクセスを許可することで、Amazon S3 オリジンを保護できるようになりました。	2022 年 8 月 24 日

CloudFront ディストリビューション用の HTTP/3 サポート	CloudFront ディストリビューションに HTTP/3 を選択できるようになりました。	2022 年 8 月 15 日
CloudFront-Viewer-TLS ヘッダーにハンドシェイクの詳細を追加	使用している SSL/TLS ハンドシェイクに関する情報を表示できるようになりました。	2022 年 6 月 27 日
Server-Timing ヘッダーの新しいメトリクス	`Server-Timing` ヘッダーに新しい `cdn-downstream-fbl` メトリクスを追加しました。	2022 年 6 月 13 日
TLS のバージョンと暗号に関する情報を取得するための新しいヘッダー	`CloudFront-Viewer-TLS` ヘッダーを使用して、TLS (または SSL) のバージョンおよびビューワーと CloudFront 間の接続に使用された暗号に関する情報を取得できるようになりました。	2022 年 5 月 23 日
CloudFront Functions の新しい FunctionThrottles メトリクス	Amazon CloudWatch を使用して、CloudFront Functions が特定の期間中にスロットリングされた回数をモニタリングできるようになりました。	2022 年 5 月 4 日
CloudFront が Lambda 関数 URL をサポート	関数 URL を持つ Lambda 関数を使用することによってサーバーレスウェブアプリケーションを構築する場合、CloudFront を追加して多くの利点を享受できるようになりました。	2022 年 4 月 6 日

HTTP レスポンスの Server-Timing ヘッダー	これで、CloudFront から送信された HTTP レスポンスの `Server-Timing` ヘッダーを使用して、CloudFront の動作とパフォーマンスに関するインサイトを得るのに役立つメトリクスを表示できます。	2022 年 3 月 30 日
AWS マネージドプレフィックスのリストを使用してインバウンドトラフィックを制限	オリジンへのインバウンド HTTP および HTTPS トラフィックの送信元を、CloudFront のオリジン向けサーバーに属する IP アドレスのみに制限できるようになりました。	2022 年 2 月 7 日
新機能	CloudFront で、[response headers policies] (レスポンスヘッダーポリシー) のサポートを追加することにより、CloudFront でビューワー (ウェブブラウザまたは他のクライアント) に送信する HTTP レスポンスに追加する HTTP ヘッダーを指定できます。オリジンを変更、またはコードを書き込むことなく、希望するヘッダー (およびその値) を指定できます。詳細については、「CloudFront レスポンスへの HTTP ヘッダーの追加または削除」を参照してください。	2021 年 11 月 2 日

新しい CloudFront-Viewer-Address リクエストヘッダー	CloudFront で、新しいヘッダー (CloudFront に HTTP リクエストを送信したビューワーの IP アドレスが含まれる CloudFront-Viewer-Address) のサポートを追加します。詳細については、「CloudFront リクエストヘッダーの追加」を参照してください。	2021 年 10 月 25 日
Lambda@Edge は新しいランタイムバージョンをサポートするようになりました	Lambda@Edge は、Lambda 関数を Python 3.9 ランタイムでサポートするようになりました。詳細については、「サポートされているランタイム」を参照してください。	2021 年 9 月 22 日
AWS マネージドポリシーの更新	CloudFront が CloudFront ReadOnlyAccess ポリシーを更新しました。詳細については、「AWS マネージドポリシーに対する CloudFront の更新」を参照してください。	2021 年 9 月 8 日
新機能	CloudFront では、ビューワー向けの HTTPS 接続の ECDSA 証明書がサポートされるようになりました。詳細については、「ビューワーと CloudFront の間でサポートされているプロトコルと暗号」および「CloudFront で SSL/TLS 証明書を使用するための要件」を参照してください。	2021 年 7 月 14 日

新機能	CloudFront では、1 つのディストリビューションから別のディストリビューションに代替ドメイン名を移動する方法について、サポート対象を拡大することになり、AWS Support までお問い合わせいただく必要がなくなりました。詳細については、「代替ドメイン名を別のディストリビューションに移動する」を参照してください。	2021 年 7 月 7 日
新しいセキュリティポリシー	CloudFront で新しいセキュリティポリシー TLSv1.2_2021 がサポートされ、サポートされる暗号のセットが小さくなりました。詳細については、「ビューワーと CloudFront との間でサポートされているプロトコルと暗号」を参照してください。	2021 年 6 月 23 日
新機能	Amazon CloudFront が CloudFront Functions をサポートするようになりました。これは、大規模でレイテンシーの影響を受けやすい CDN カスタマイズのために軽量な関数を JavaScript で記述することを可能にする CloudFront のネイティブ機能です。詳細については、「CloudFront Functions を使用したエッジでのカスタマイズ」を参照してください。	2021 年 5 月 3 日

Lambda@Edge は新しいランタイムバージョンをサポートします	Lambda@Edge は、Lambda 関数を Node.js 14 ランタイムでサポートするようになりました。詳細については、「サポートされているランタイム」を参照してください。	2021 年 4 月 29 日
RTMP ディストリビューションのドキュメントを削除する	Amazon CloudFront は 2020 年 12 月 31 日に、リアルタイムメッセージングプロトコル (RTMP) ディストリビューションを非推奨にしました。RTMP ディストリビューションのドキュメントは Amazon CloudFront デベロッパーガイドから削除されます。	2021 年 2 月 10 日
新しい料金オプション	Amazon CloudFront が CloudFront Security Savings Bundle を導入します。これは、AWS 請求書の CloudFront 料金を最大 30% 削減できる簡単な方法です。詳細については、よくある質問の「Savings Bundle」を参照してください。	2021 年 2 月 5 日

チュートリアルの新規追加	Amazon CloudFront デベロッパーガイドには、Amazon CloudFront を使用して Elastic Load Balancing で Application Load Balancer へのアクセスを制限するためのチュートリアルが含まれています。詳細については、「Application Load Balancer へのアクセスを制限する」を参照してください。	2020 年 12 月 18 日
パブリックキー管理の新しいオプション	CloudFront は、AWS アカウント ルートユーザーへのアクセスを必要とせずに、CloudFront コンソールと API を介して署名付き URL と署名付き Cookie のパブリックキー管理をサポートするようになりました。詳細については、「署名付き URL と署名付き Cookie を作成できる署名者の指定」を参照してください。	2020 年 10 月 22 日
新機能 – Origin Shield	CloudFront は CloudFront Origin Shield をサポートするようになりました。これは、オリジンの負荷を最小限に抑え、可用性を向上させ、運用コストの削減に役立つ、CloudFront キャッシュインラストラクチャ内の追加レイヤーです。詳細については、「Amazon CloudFront Origin Shield の使用」を参照してください。	2020 年 10 月 20 日

新しい圧縮形式	CloudFront では、CloudFront のエッジロケーションでオブジェクトが圧縮されるように CloudFront を設定した場合に、Brotli 圧縮形式がサポートされるようになりました。正規化された `Accept-Encoding` ヘッダーを使用して Brotli オブジェクトがキャッシュされるように、CloudFront を設定することもできます。詳細については、「圧縮ファイルの供給」と「圧縮サポート」を参照してください。	2020 年 9 月 14 日
新しい TLS プロトコル	CloudFront では、ビューワーと CloudFront ディストリビューションの間の HTTPS 接続に対して TLS 1.3 プロトコルがサポートされるようになりました。すべての CloudFront セキュリティポリシーで、デフォルトでは TLS 1.3 が有効になっています。詳細については、「ビューワーと CloudFront との間でサポートされているプロトコルと暗号」を参照してください。	2020 年 9 月 3 日

新しいリアルタイムログ	CloudFront で設定可能なリアルタイムログがサポートされるようになりました。リアルタイムログを使用して、ディストリビューションに対して行われたリクエストに関する情報をリアルタイムで取得できます。リアルタイムログを使用して、コンテンツ配信のパフォーマンスに基づいて監視、分析、アクションを実行できます。詳細については、「リアルタイムログ」を参照してください。	2020 年 8 月 31 日
追加のメトリクスの API サポート	CloudFront で、CloudFront API を使用した 8 つのリアルタイムメトリクスを追加で有効にできるようになりました。詳細については、「追加のメトリクスを有効にする」を参照してください。	2020 年 8 月 28 日
新しい CloudFront HTTP ヘッダー	CloudFront では、デバイスタイプ、地理的位置などのビューワーに関する情報を決定するために追加の HTTP ヘッダーが追加されました。詳細については、「CloudFront リクエストヘッダーの追加」を参照してください。	2020 年 7 月 23 日

新機能	CloudFront で、[cache policies] (キャッシュポリシー) と [origin request polices] (オリジンリクエストポリシー) をサポートするようになりました。これにより、CloudFront ディストリビューションの [cache key] (キャッシュキー) と [origin requests] (オリジンリクエスト) をさらにきめ細かく制御できるようになります。詳細については、「キャッシュキーの管理」および「オリジンリクエストの制御」を参照してください。	2020 年 7 月 22 日
新しいセキュリティポリシー	CloudFront で新しいセキュリティポリシー TLSv1.2_2019 がサポートされ、サポートされる暗号のセットが小さくなりました。詳細については、「ビューワーと CloudFront との間でサポートされているプロトコルと暗号」を参照してください。	2020年7月8日
オリジンのタイムアウトと試行を制御する新しい設定	CloudFront には、オリジンのタイムアウトと試行を制御する新しい設定が追加されています。詳細については、「オリジンタイムアウトと試行の制御」を参照してください。	2020 年 6 月 5 日

安全な静的ウェブサイトを作成して CloudFront の使用を開始するための新しいドキュメント	Amazon S3、CloudFront、Lambda@Edge などを使用してセキュアな静的ウェブサイトを作成して、CloudFront の使用を開始しましょう。これらはすべて AWS CloudFormation を使用してデプロイされます。詳細については、「安全な静的ウェブサイトの使用開始」を参照してください。	2020 年 6 月 2 日
Lambda@Edge は新しいランタイムバージョンをサポートします	Lambda@Edge は、Lambda 関数を Node.js 12 および Python 3.8 ランタイムでサポートするようになりました。詳細については、「サポートされているランタイム」を参照してください。	2020 年 2 月 27 日
CloudWatch の新しいリアルタイムメトリクス	Amazon CloudFront は、Amazon CloudWatch の 8 つの追加のリアルタイムメトリクスを提供するようになりました。詳細については、「追加の CloudFront ディストリビューションメトリクスを有効にする」を参照してください。	2019 年 12 月 19 日
アクセスログの新しいフィールド	CloudFront に、ログにアクセスするための 7 つの新しいフィールドが追加されました。詳細については、「標準ログファイルフィールド」を参照してください。	2019 年 12 月 12 日

AWS WordPress プラグイン	AWS プラグインを使用して、WordPress ウェブサイトへの訪問者に CloudFront を使用した高速表示エクスペリエンスを提供できます。(更新; 2022 年 9 月 30 日現在、WordPress 用の AWS プラグインは非推奨になっています)	2019 年 10 月 30 日
タグベースとリソースレベルの IAM アクセス許可ポリシー	CloudFront は、IAM アクセス許可ポリシーを指定する 2 つの追加の方法として、タグベースとリソースレベルのアクセス許可ポリシーをサポートするようになりました。詳細については、「リソースへのアクセスの管理」を参照してください。	2019 年 8 月 8 日
Python プログラミング言語のサポート	Lambda@Edge で、Node.js に加えて Python プログラミング言語でも関数を開発できるようになりました。さまざまなシナリオに対応する関数の例については、「Lambda@Edge 関数の例」を参照してください。	2019 年 8 月 1 日

更新されたモニタリンググラフ	CloudFront コンソールから直接 CloudFront ディストリビューションに関連する Lambda 関数を監視して、エラーをより簡単に追跡およびデバッグするための新しい方法を説明するためのコンテンツの更新。詳細については、「CloudFront のモニタリング」を参照してください。	2013 年 6 月 20 日
一括セキュリティコンテンツ	新しいセキュリティの章では、CloudFront のデータ保護、IAM、ロギング、コンプライアンスなどの機能と実装に関する情報を統合しています。詳細については、「セキュリティ」を参照してください。	2019 年 5 月 24 日
ドメイン検証が必要になりました	CloudFront では、ディストリビューションで代替ドメイン名を使用するアクセス権限があることを確認する SSL 証明書を使用することが必要になりました。詳細については、「代替ドメイン名と HTTPS の使用」を参照してください。	2019 年 4 月 9 日
更新された PDF ファイル名	Amazon CloudFront 開発者ガイドの新しいファイル名は、AmazonCloudFront_DevGuide です。以前の名前は cf-dg でした。	2019 年 1 月 7 日

新しい特徴	CloudFront は、WebSocket をサポートするようになりました。これは、クライアントとサーバー間の長時間接続が必要な場合に便利な TCP ベースのプロトコルです。高可用性が必要なシナリオでは、オリジンフェイルオーバーを使用して CloudFront を設定することもできるようになりました。詳細については、「CloudFront ディストリビューションで WebSocket を使用する」および「CloudFront オリジンフェイルオーバーによる高可用性の最適化」を参照してください。	2018 年 11 月 20 日
新機能	CloudFront では、Lambda 関数を実行する HTTP リクエストの詳細エラーのログ記録をサポートするようになりました。ログを CloudWatch に保存すると、関数から無効なレスポンスが返された際に、保存したログを使用して HTTP 5xx エラーをトラブルシューティングすることができます。詳細については、「Lambda 関数の CloudWatch メトリクスと CloudWatch Logs」を参照してください。	2018 年 10 月 8 日

新機能	書き込み可能な HTTP メソッド (POST、PUT、DELETE など) のリクエストのボディを Lambda@Edge で公開することを選択できるようになり、Lambda 関数でそのボディにアクセスできます。読み取り専用アクセスを選択することも、ボディを置き換えることを指定することもできます。詳細については、「Include Body オプションの選択によるリクエストボディへのアクセス」を参照してください。	2018 年 8 月 14 日
新機能	CloudFront は、gzip に加えて、または gzip の代わりに、brotli または他の圧縮アルゴリズムを使用して圧縮されたコンテンツを提供するようになりました。詳細については、「圧縮ファイルの供給」を参照してください。	2018 年 7 月 25 日
再構成	Amazon CloudFront 開発者ガイドは、関連したコンテンツの検索を簡単にし、スキャン性能とナビゲーションを向上させるために再編成されました。	2018 年 28 月 6 日

新機能	オリジン側イベント内で、追加のヘッダー (カスタムヘッダーを含む) にアクセスできるようになったことで、Lambda@Edge を使用して、Amazon S3 バケットに保存されたコンテンツの配信をさらにカスタマイズできるようになりました。詳細については、ビューワーの場所およびビューワーのデバイスタイプに基づいたコンテンツのパーソナライズを示す例を参照してください。	2018 年 3 月 20 日
新機能	Amazon CloudFront を使用して、楕円曲線 DSA (ECDSA) を使用したオリジンへの HTTPS 接続をネゴシエートできるようになりました。ECDSA は旧 RSA アルゴリズムと比較してより小さいキーを使用し、より高速でありながら、安全性は同等です。詳細については、「CloudFront とオリジン間の通信用にサポートされる SSL/TLS プロトコルと暗号」および「RSA および ECDSA 暗号について」を参照してください。	2018 年 3 月 15 日

新機能	Lambda@Edge を使用すると、Amazon CloudFront がオリジンから受信した HTTP エラーに応答して Lambda 関数を実行できるため、オリジンからのエラーレスポンスをカスタマイズできるようになりました。詳細については、別のロケーションへのリダイレクトの例および 200 ステータスコード (OK) を伴ったレスポンス生成の例を参照してください。	2017 年 21 月 12 日
新機能	新しい CloudFront 機能であるフィールドレベル暗号化を使用して、クレジットカード番号や社会保障番号のように個人を識別できる情報 (PII) など、機密データのセキュリティをより強化できるようになりました。詳細については、「フィールドレベル暗号化を使用した機密データの保護」を参照してください。	2017 年 14 月 12 日
ドキュメント履歴のアーカイブ	古いドキュメント履歴がアーカイブされました。	2017 年 12 月 1 日