開発者ガイド

AWS SDK for PHP

AWS SDK for PHP: 開発者ガイド

Table of Contents

AWS SDK for PHP バージョン 3 とは

AWS SDK for PHP バージョン 3 により、PHP デベロッパーは PHP コードで Amazon Web Services を使用し、Amazon S3、Amazon DynamoDB、S3 Glacier などのサービスを使用して堅牢なアプリケーションやソフトウェアを構築できます。Composer を使用して `aws/aws-sdk-php` パッケージをインクルードするか、スタンドアロンの `aws.zip` または `aws.phar` ファイルをダウンロードすることによって、この SDK をインストールすると、数分で使用開始できます。

SDK では、一部のサービスはすぐには使用できません。AWS SDK for PHP で現在サポートされているサービスを見つけるには、「Service Name and API Version」を参照してください。

> **Note**
>
> SDK バージョン 2 を使用しているコードを SDK バージョン 3 に移行する場合は、「AWS SDK for PHP のバージョン 2 からのアップグレード」を必ずお読みください。

SDK の使用を開始する

SDK を実際に使用する準備ができている場合は、使用を開始する の章に従ってください。AWS による認証、開発環境のセットアップ、Amazon S3 を使用した最初の基本アプリケーションの作成について順を追って説明します。

その他のリソース

- よくある質問
- 用語集
- AWS SDK およびツールリファレンスガイドには、AWS SDK に共通する設定、機能、その他の基本概念も含まれています。
- Guzzle のドキュメント
- AWS SDK for PHP を使用したコード例は、awsdocs/aws-doc-sdk-examples リポジトリにあります。
- Gitter の PHP SDK コミュニティ。
- AWS re:Post.

GitHub:

- AWS SDK for PHP のソースコードは aws/aws-sdk-php リポジトリにあります。
- この SDK への貢献
- バグを報告するか、機能をリクエストする

API ドキュメント

SDK の API ドキュメントは https://docs.aws.amazon.com/sdk-for-php/latest/reference/ にあります。

SDK メジャーバージョンのメンテナンスとサポート

SDK メジャーバージョンのメンテナンスとサポート、およびその基礎的な依存関係については、AWS SDK とツール共有設定および認証情報リファレンスガイドで以下を参照してください。

- AWS SDK とツールのメンテナンスポリシー
- AWS SDK とツールのバージョンサポートマトリクス

使用を開始する

この章では、 AWS SDK for PHP バージョン 3 を起動して実行する方法について説明します。

トピック

- での SDK 認証 AWS
- AWS SDK for PHP バージョン 3 の要件と推奨事項
- AWS SDK for PHP バージョン 3 のインストール
- AWS SDK for PHP の Hello チュートリアル
- AWS Cloud9 を AWS SDK for PHP と併用する

での SDK 認証 AWS

で開発 AWS する場合、コードが でどのように認証されるかを確立する必要があります AWS サービス。 AWS リソースへのプログラムによるアクセスは、環境や利用可能な AWS アクセスに応じて、さまざまな方法で設定できます。

認証方法を選択して SDK 用に設定するには、AWS SDK とツールのリファレンスガイドの「認証とアクセス」を参照してください。

ローカルで開発していて、雇用主から認証方法が与えられていない新しいユーザーは、 をセットアップすることをお勧めします AWS IAM Identity Center。この方法には、設定を簡単に AWS CLI するために をインストールしたり、 AWS アクセスポータルに定期的にサインインしたりすることが含まれます。この方法を選択した場合、AWS SDK とツールのリファレンスガイドの IAM Identity Center 認証の手順を完了したあと、環境には次の要素が含まれるはずです。

- アプリケーションを実行する前に AWS アクセスポータルセッションを開始 AWS CLIするために使用する 。
- SDK が参照できる設定値のセットを含む`[default]`プロファイルを持つ共有 AWS`config`ファイル。このファイルの場所を確認するには、AWS SDK とツールのリファレンスガイドの「共有ファイルの場所」を参照してください。
- 共有`config`ファイルには `region`設定が含まれています。これにより、SDK AWS リージョン がリクエストに使用するデフォルトが設定されます。このリージョンは、 `region`プロパティで明示的に設定されていない SDK サービスリクエストに使用されます。

- SDK は、リクエストを AWSに送信する前に、プロファイルの SSO トークンプロバイダー設定を使用して認証情報を取得します。IAM Identity Center アクセス許可セットに接続された IAM ロールである sso_role_name値により、アプリケーションで AWS サービス 使用されている へのアクセスが許可されます。

  次のサンプル config ファイルは、SSO トークンプロバイダー設定で設定されたデフォルトプロファイルを示しています。プロファイルの sso_session 設定は、指定された sso-session セクションを参照します。sso-session セクションには、 AWS アクセスポータルセッションを開始するための設定が含まれています。

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
region = us-east-1
output = json

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://provided-domain.awsapps.com/start
sso_registration_scopes = sso:account:access
```

IAM Identity Center 認証を使用するために、アプリケーションに追加のパッケージ (SSOや などSSOOIDC) AWS SDK for PHP は必要ありません。

AWS アクセスポータルセッションを開始する

にアクセスするアプリケーションを実行する前に AWS サービス、SDK が IAM Identity Center 認証を使用して認証情報を解決するためのアクティブな AWS アクセスポータルセッションが必要です。設定したセッションの長さによっては、アクセスが最終的に期限切れになり、SDK で認証エラーが発生します。 AWS アクセスポータルにサインインするには、 で次のコマンドを実行します AWS CLI。

```
aws sso login
```

ガイダンスに従い、デフォルトのプロファイルを設定している場合は、--profile オプションを指定してコマンドを呼び出す必要はありません。SSO トークンプロバイダー設定で名前付きプロファイルを使用している場合、コマンドは aws sso login --profile named-profile です。

既にアクティブなセッションがあるかどうかをオプションでテストするには、次の AWS CLI コマンドを実行します。

```
aws sts get-caller-identity
```

セッションがアクティブな場合、このコマンドへの応答により、共有 config ファイルに設定されている IAM Identity Center アカウントとアクセス許可のセットが報告されます。

> **Note**
>
> 既にアクティブな AWS アクセスポータルセッションがあり、 を実行している場合はaws sso login、 認証情報を提供する必要はありません。
> サインインプロセスでは、データ AWS CLI へのアクセスを許可するように求められる場合があります。 AWS CLI は SDK for Python 上に構築されているため、アクセス許可メッセージにはbotocore名前のバリエーションが含まれている場合があります。

認証の詳細はこちら

- 認証に IAM Identity Center を使用する方法の詳細については、「 SDK とツールのリファレンスガイド」の「IAM Identity Center 認証を理解する」を参照してください。 AWS SDKs
- 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。
- 短期 AWS 認証情報を作成するには、「IAM ユーザーガイド」の「一時的なセキュリティ認証情報」を参照してください。
- が AWS SDK for PHP 使用できるその他の認証情報プロバイダーについては、「 SDK およびツールリファレンスガイド」の「標準化された認証情報プロバイダー」を参照してください。 AWS SDKs

AWS SDK for PHP バージョン 3 の要件と推奨事項

AWS SDK for PHP で最良の結果を得るには、使用する環境で以下の要件と推奨事項がサポートされていることを確認します。

要件

AWS SDK for PHP を使用するには、PHP バージョン 5.5.0 以降を使用し、SimpleXML PHP 拡張機能を有効にする必要があります。Amazon CloudFront のプライベート URL に署名する必要がある場合は、OpenSSL PHP 拡張機能も必要です。

レコメンデーション

最小要件に加えて、以下をインストール、アンインストール、および使用することもお勧めします。

cURL 7.16.2 以降をインストールする	OpenSSL/NSS および zlib でコンパイルされている最新バージョンの cURL を使用します。cURL がシステムにインストールされていなくて、クライアント用のカスタム http_handler を設定していない場合、SDK では PHP ストリームラッパーが使用されます。
OPCache を使用する	OPcache 拡張機能を使用して、事前コンパイルされたスクリプトバイトコードを共有メモリに格納することによって、PHP のパフォーマンスが向上します。そうすることにより、各リクエストに対して PHP がスクリプトをロードして解析する必要がなくなります。この拡張機能は、通常はデフォルトで有効になっています。 Amazon Linux を実行している場合に OPCache 拡張機能を使用するには、php56-opcache または php55-opcache の yum パッケージをインストールする必要があります。
実稼働環境で Xdebug をアンインストールする	Xdebug はパフォーマンスのボトルネックの特定に役立ちます。ただし、パフォーマンスを重視するアプリケーションの場合は、本稼働環境に Xdebug 拡張機能をインストールしないでください。この拡張機能をロードすると、SDK のパフォーマンスが大幅に低下します。

Composer の classmap 自動ローダーを使用する	自動ローダーは、PHP スクリプトで必要になったときにクラスをロードします。Composer は、アプリケーションの PHP スクリプトおよびアプリケーションで必要な他のすべての PHP スクリプト (AWS SDK for PHP を含む) を自動ロードできる自動ローダーを生成します。 本稼働環境では、自動ローダーのパフォーマンスを向上させるために classmap 自動ローダーを使用することをお勧めします。classmap 自動ローダーは、Composer のインストールコマンドに `-o` または `==optimize-autoloader` オプションを渡すことによって生成できます。

互換性テスト

SDK コードベースにある `compatibility-test.php` ファイルを実行して、使用するシステムで SDK を実行できることを確認します。互換性テストでは、SDK の最小システム要件を満たしていることだけでなく、オプションの設定もチェックされ、パフォーマンスを向上させるのに役立つ推奨事項が提示されます。互換性テストでは、コマンドラインまたはウェブブラウザのいずれかに結果が出力されます。テスト結果をブラウザで確認する場合、成功したチェックは緑色で、警告は紫色で、失敗は赤色で表示されます。コマンドラインで実行した場合は、項目ごとのチェック結果が個別の行に出力されます。

SDK に関する問題がレポートされている場合は、互換性テストの結果を共有すると、根本的な原因の特定に役立ちます。

AWS SDK for PHP バージョン 3 のインストール

AWS SDK for PHP バージョン 3 は、次の方法でインストールできます。

- Composer を介して依存関係として
- SDK のパッケージ済み phar として
- SDK の ZIP ファイルとして

AWS SDK for PHP バージョン 3 をインストールする前に、環境が PHP バージョン 5.5 以降を使用していることを確認します。環境の要件と推奨事項の詳細を確認してください。

> **Note**
>
> .phar および .zip 形式で SDK をインストールする場合は、マルチバイト文字列 PHP 拡張機能を別途インストールして有効にする必要があります。

Composer を介して依存関係として AWS SDK for PHP をインストールする

AWS SDK for PHP をインストールするには Composer を使用する方法をお勧めします。Composer は、プロジェクトの依存関係を管理およびインストールする PHP 用のツールです。

Composer のインストール方法、自動ロードの設定方法、および依存関係定義の他のベストプラクティスに従う方法の詳細については、getcomposer.org を参照してください。

Composer をインストールする

Composer がまだプロジェクトにない場合は、Composer をダウンロードページで Composer をダウンロードしてインストールします。

- Windows の場合、Windows Installer の指示に従ってください。
- Linux の場合、コマンド行インストールの指示に従います。

Composer を介して依存関係として AWS SDK for PHP を追加する

Composer がシステムにグローバルにインストール済みの場合、プロジェクトのベースディレクトリで以下の操作を実行して、依存関係として AWS SDK for PHP をインストールします。

```
$ composer require aws/aws-sdk-php
```

それ以外の場合は、この Composer コマンドを入力し、依存関係として AWS SDK for PHP の最新バージョンをインストールします。

```
$ php -d memory_limit=-1 composer.phar require aws/aws-sdk-php
```

php スクリプトに自動ローダーを追加する

Composer をインストールすると、複数のフォルダとファイルが環境に作成されます。最初に使用するファイルは `autoload.php` です。これは 環境 の `vendor` フォルダにあります。

スクリプトで AWS SDK for PHP を活用するには、次のようにスクリプトに自動ローダーを追加します。

```
<?php
   require '/path/to/vendor/autoload.php';
?>
```

パッケージ済み phar を使用したインストール

AWS SDK for PHP の各リリースには、SDK を実行するために必要なクラスと依存関係がすべて含まれているパッケージ済みの phar (PHP アーカイブ) が付属しています。また、その phar ファイルでは、AWS SDK for PHP 用のクラスの自動ローダーとそのすべての依存関係が自動的に登録されます。

パッケージ済み phar をダウンロードして、スクリプトでインクルードすることができます。

```
<?php
   require '/path/to/aws.phar';
?>
```

> **Note**
>
> Suhosin パッチで PHP を使用することはお勧めしませんが、Ubuntu や Debian ディストリビューションではよく使用されています。そうする場合は、suhosin.ini で phar の使用を有効にする必要があります。有効にしていない場合に、コード内で phar ファイルをインクルードすると、エラーが表示されずに失敗します。有効にするには、suhosin.ini に次の行を追加します。
>
> ```
> suhosin.executor.include.whitelist = phar
> ```

ZIP ファイルを使用したインストール

AWS SDK for PHP には、SDK を実行するために必要なクラスと依存関係がすべて含まれている ZIP ファイルが付属しています。また、その ZIP ファイルには、AWS SDK for PHP 用のクラスの自動ローダーとその依存関係が含まれています。

SDK をインストールするには、.zip ファイルをダウンロードし、プロジェクト内の任意の場所に展開します。次に、スクリプトで次のように自動ローダーをインクルードします。

```
<?php
   require '/path/to/aws-autoloader.php';
?>
```

AWS SDK for PHP の Hello チュートリアル

AWS SDK for PHP を使用して Amazon S3 に挨拶してください。次の例では、すべての Amazon S3 バケットのリストを表示します。

コードで SDK をインクルードする

どのような手法で SDK でインストールにしたかにかかわらず、1 つの `require` ステートメントだけで SDK をインクルードできます。インストール手法に最適な PHP コードについては、次の表を参照してください。`/path/to/` は、使用しているシステムでの実際のパスに置き換えます。

インストール手法	require ステートメント
Composer の使用	`require '/path/to/vendor/autoload.php';`
phar の使用	`require '/path/to/aws.phar';`
ZIP の使用	`require '/path/to/aws-autoloader.php';`

このトピックでは、Composer のインストール方法を示します。別のインストール方法を使用している場合は、このセクションを参照して適切な `require` コードを見つけてください。

コードを書き込む

次のコードをコピーし、新しいソースファイルに貼り付けます。ファイルを保存して hello-s3.php という名前を付けます。

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;


/**
 * List your Amazon S3 buckets.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

//Create a S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

//Listing all S3 Bucket
$buckets = $s3Client->listBuckets();
foreach ($buckets['Buckets'] as $bucket) {
    echo $bucket['Name'] . "\n";
}
```

プログラムの実行

PHP プログラムを実行するには、コマンドプロンプトを開きます。PHP プログラムを実行する一般的なコマンド構文は次のとおりです。

```
php [source filename] [arguments...]
```

このサンプルコードは引数を使用しません。このコードを実行するには、コマンドプロンプトで以下を入力します。

```
$ php hello-s3.php
```

次のステップ

他の多くの Amazon S3 オペレーションをテストするには、「」のAWS「コード例リポジトリ」を確認してください GitHub。

AWS Cloud9 を AWS SDK for PHP と併用する

AWS Cloud9 は、ウェブベースの統合開発環境 (IDE) であり、クラウド内のソフトウェアのコーディング、ビルド、テスト、デバッグ、リリースに使用する一連のツールが含まれています。AWS SDK for PHP で AWS Cloud9 を使用すると、ブラウザを使って PHP コードを作成して実行することができます。AWS Cloud9 には、コードエディタやターミナルなどのツールが含まれています。AWS Cloud9 IDE はクラウドベースであるため、インターネットに接続されたマシンを使用して、オフィス、自宅、その他どこからでもプロジェクトに取り組むことができます。AWS Cloud9 に関する一般的な情報については、AWS Cloud9ユーザーガイドを参照してください。

以下の手順に従って AWS SDK for PHP で AWS Cloud9 を設定してください。

- ステップ 1: AWS Cloud9 を使用するために AWS アカウント をセットアップする
- ステップ 2: AWS Cloud9 開発環境を設定する
- ステップ 3: AWS SDK for PHP を設定する
- ステップ 4: サンプルコードをダウンロードする
- ステップ 5: コードサンプルを実行する

ステップ 1: AWS Cloud9 を使用するために AWS アカウント をセットアップする

AWS Cloud9 を使用するには、AWS Management Console から AWS Cloud9 コンソールにサインインします。

> **Note**
>
> AWS IAM Identity Center を使用して認証を行う場合、IAM `iam:ListInstanceProfilesForRole` コンソールのユーザーアタッチポリシーにの必要な権限を追加する必要があるかもしれません。

AWS Cloud9 にアクセスして AWS Cloud9 コンソールにサインインするように AWS アカウントで IAM エンティティをセットアップするには、AWS Cloud9ユーザーガイドの AWS Cloud9 のチームのセットアップを参照してください。

ステップ 2: AWS Cloud9の開発環境を設定

AWS Cloud9 コンソールにサインインしたら、コンソールを使用して AWS Cloud9 開発環境を作成します。環境を作成すると、その環境の AWS Cloud9 IDE が表示されます。

詳細については、AWS Cloud9 ユーザーガイドの「AWS Cloud9 での環境の作成」を参照してください。

> **Note**
>
> コンソールで始めて環境を作成する際に、[Create a new instance for environment (EC2)] (環境の新しいインスタンスを作成する) (EC2) オプションを選択することをお勧めします。このオプションは、環境を作成し、Amazon EC2 インスタンスを起動した後、新しいインスタンスを新しい環境に接続するように AWS Cloud9 に指示します。これは、AWS Cloud9 を使用するための最も手軽な方法です。

IDE でターミナルが開いていない場合は開きます。IDE のメニューバーで、[Window]、[New Terminal] の順に選択します。ターミナルウィンドウを使用してツールをインストールし、アプリケーションを構築できます。

ステップ 3: AWS SDK for PHP をセットアップする

AWS Cloud9 が開発環境の IDE を開いたら、ターミナルウィンドウを使用して、以下のように環境内で AWS SDK for PHP を設定します。

AWS SDK for PHP をインストールするには Composer を使用する方法をお勧めします。Composer は、プロジェクトの依存関係を管理およびインストールする PHP 用のツールです。

Composer のインストール方法、自動ロードの設定方法、および依存関係定義の他のベストプラクティスに従う方法の詳細については、getcomposer.org を参照してください。

Composer をインストールする

Composer がまだプロジェクトにない場合は、Composer をダウンロードページで Composer をダウンロードしてインストールします。

- Windows の場合、Windows Installer の指示に従ってください。
- Linux の場合、コマンド行インストールの指示に従います。

Composer を介して依存関係として AWS SDK for PHP を追加する

Composer がシステムにグローバルにインストール済みの場合、プロジェクトのベースディレクトリで以下の操作を実行して、依存関係として AWS SDK for PHP をインストールします。

```
$ composer require aws/aws-sdk-php
```

それ以外の場合は、この Composer コマンドを入力し、依存関係として AWS SDK for PHP の最新バージョンをインストールします。

```
$ php -d memory_limit=-1 composer.phar require aws/aws-sdk-php
```

php スクリプトに自動ローダーを追加する

Composer をインストールすると、複数のフォルダとファイルが環境に作成されます。最初に使用するファイルは `autoload.php` です。これは 環境 の `vendor` フォルダにあります。

スクリプトで AWS SDK for PHP を活用するには、次のようにスクリプトに自動ローダーを追加します。

```
<?php
   require '/path/to/vendor/autoload.php';
?>
```

ステップ 4: サンプルコードをダウンロード

ターミナルウィンドウを使用して、AWS SDK for PHP のコードサンプルを AWS Cloud9 開発環境にダウンロードします。

AWS SDK の公式ドキュメントで使用されているすべてのサンプルコードのコピーをダウンロードするには、次のコマンドを実行します。

```
$ git clone https://github.com/awsdocs/aws-doc-sdk-examples.git
```

AWS SDK for PHP のコードサンプルは `ENVIRONMENT_NAME/aws-doc-sdk-examples/php` ディレクトリにあります。ここで、`ENVIRONMENT_NAME` は開発環境の名前です。

Amazon S3 の例を使用して進めるには、コードサンプル `ENVIRONMENT_NAME/aws-doc-sdk-examples/php/example_code/s3/ListBuckets.php` から始めることをお勧めします。この例では、すべての Amazon S3 バケットを一覧表示します ターミナルウィンドウを使用して `s3` ディレクトリに移動し、ファイルを一覧表示します。

```
$ cd aws-doc-sdk-examples/php/example_code/s3
$ ls
```

AWS Cloud9 でファイルを開くには、ターミナルウィンドウで直接を `ListBuckets.php` ディレクトリをクリックします。

コードサンプルについてさらに理解を深めるには、「AWS SDK for PHP コードサンプル」を参照してください。

ステップ 5: コードサンプルの実行

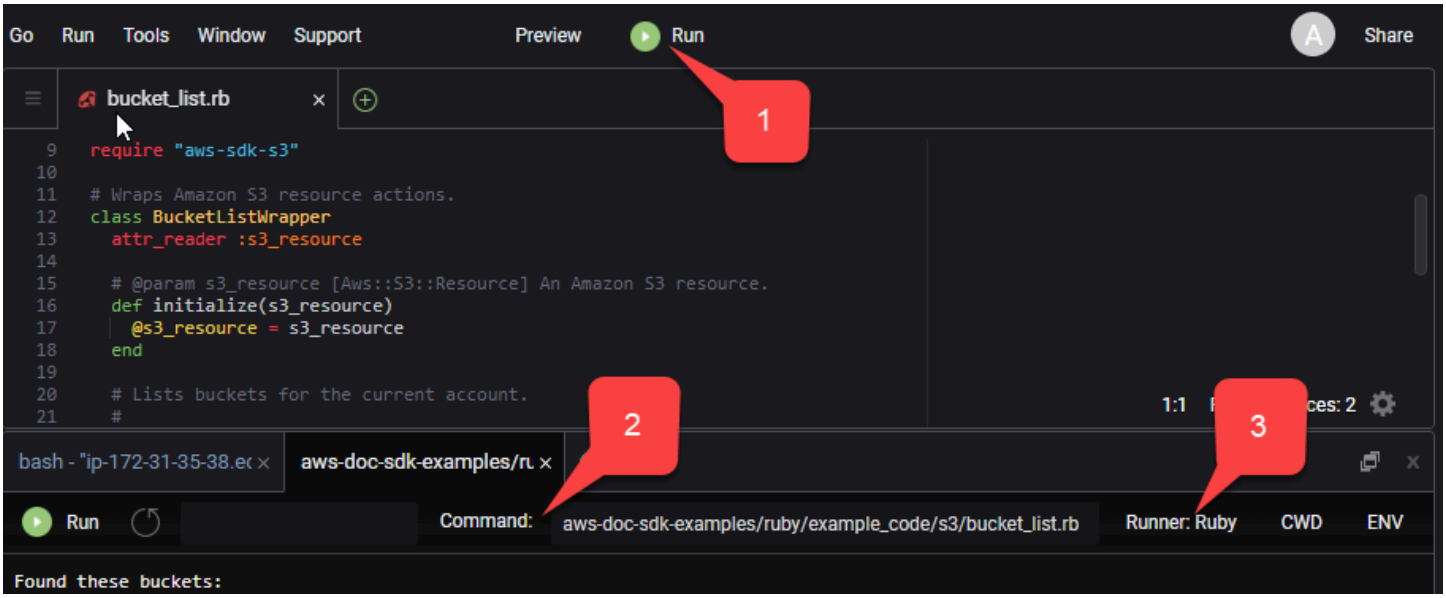
AWS Cloud9 開発環境でコードを実行するには、上部のメニューバーの [実行] ボタンを選択します。AWS Cloud9 は `.php` ファイル拡張子を自動的に検出し、PHP (ビルトイン Web サーバー) ランナーを使用してコードを実行します。ただし、この例では実際には PHP (**`cli`**) オプションが必要です。AWS Cloud9 でコードを実行する際の詳細については、AWS Cloud9 ユーザーガイドの「コードを実行する」を参照してください。

以下のスクリーンショットでは、次の基本的な点に注意してください。

- 1: 実行します。[実行] ボタンは上部のメニューバーにあります。新しいタブが開き、結果が表示されます。

  > **Note**
  >
  > 新しい実行設定を手動で作成することもできます。メニューバーで、 [Run (実行)] 、 [Run Configurations (実行設定)] 、 [New Run Configuration (新しい実行設定)] の順に選択します。

- 2: コマンド。AWS Cloud9コマンドテキストボックスに、実行するファイルへのパスとファイル名を入力します。コードでコマンドラインパラメータを渡す必要がある場合は、ターミナルウィンドウからコードを実行する場合と同じ方法でコマンドラインにパラメータを追加できます。
- 3: ランナー。AWS Cloud9 は、ファイル拡張子が `.php` であることを検出し、PHP (ビルトイン Web サーバー) ランナーを選択してコードを実行します。代わりにこの例を実行するには PHP (**`cli`**) を選択してください。



実行中のコードから生成された出力はすべてタブに表示されます。

AWS SDK for PHP バージョン 3 の設定

AWS SDK for PHP はさまざまな機能とコンポーネントで構成されています。以下の各トピックで、SDK で使用されているコンポーネントについて説明します。

AWS SDK とツールのリファレンスガイドには、AWS SDK の多くに共通する設定、機能、その他の基本概念も含まれています。

トピック

AWS SDK for PHP バージョン 3 の基本的な使用パターン

このトピックでは、AWS SDK for PHP の基本的な使用パターンについて説明します。

前提条件

- SDK をダウンロードしてインストールする
- AWS SDK for PHP を使用する前に、AWS で認証する必要があります。認証の設定の詳細については、「での SDK 認証 AWS」を参照してください。

コードで SDK をインクルードする

どのような手法で SDK でインストールにしたかにかかわらず、1 つの `require` ステートメントだけで SDK をインクルードできます。インストール手法に最適な PHP コードについては、次の表を参照してください。`/path/to/` は、使用しているシステムでの実際のパスに置き換えます。

インストール手法	require ステートメント
Composer の使用	`require '/path/to/vendor/autoload.php';`
phar の使用	`require '/path/to/aws.phar';`
ZIP の使用	`require '/path/to/aws-autoloader.php';`

このトピックでは、Composer のインストール方法を示します。別のインストール方法を使用している場合は、このセクションを参照して適切な `require` コードを見つけてください。

使用法の概要

SDK を使用して AWS のサービスとやり取りするには、クライアントオブジェクトをインスタンス化します。クライアントオブジェクトには、サービスの API のオペレーションと対応するメソッドがあります。特定のオペレーションを実行するには、それに対応するメソッドを呼び出します。そのメソッドでは、成功すると配列に類似した Result オブジェクトが返され、失敗すると 例外がスローされます。

クライアントの作成

クライアントのコンストラクタにオプションの連想配列を渡すことによって、クライアントを作成できます。

インポート

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;
```

サンプルコード

```
//Create an S3Client
$s3 = new Aws\S3\S3Client([
    'region' => 'us-east-2'  // Since version 3.277.10 of the SDK,
]);                          // the 'version' parameter defaults to 'latest'.
```

オプションの「version」パラメーターに関する情報は、「設定オプション」のトピックに記載されています。

このクライアントには明示的に認証情報を指定していないことがわかります。これは、SDK が環境変数、ホームディレクトリにある 共有 config および credentials ファイル 認証情報の INI ファイル、AWS Identity and Access Management (IAM) のインスタンスプロファイルの認証情報、認証情報プロバイダのいずれかから認証情報を検出するためです。

すべての一般的なクライアント設定オプションについては、「AWS SDK for PHP バージョン 3 の設定」で詳しく説明されています。クライアントに指定されるオプションの配列は、作成するクライアントによって異なります。これらのカスタムクライアント設定オプションについては、各クライアントの API ドキュメントで説明されています。

Sdk クラスの使用

Aws\Sdk クラスは、クライアントファクトリとして機能し、複数のクライアント間で共有される設定オプションを管理するために使用されます。特定のクライアントコンストラクタに指定できる多くのオプションを、Aws\Sdk クラスに対しても指定できます。それらのオプションは、各クライアントコンストラクタにも適用されます。

インポート

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;
```

サンプルコード

```
// The same options that can be provided to a specific client constructor can also be
 supplied to the Aws\Sdk class.
// Use the us-west-2 region and latest version of each client.
```

```
$sharedConfig = [
    'region' => 'us-west-2'
];
// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk($sharedConfig);
// Create an Amazon S3 client using the shared configuration data.
$client = $sdk->createS3();
```

すべてのクライアント間で共有されるオプションは、ルートレベルのキーと値のペアに配置されます。サービス固有の設定データは、サービスの名前空間 (「S3」、「DynamoDb」など) と同じキーで指定できます。

```
$sdk = new Aws\Sdk([
    'region'   => 'us-west-2',
    'DynamoDb' => [
        'region' => 'eu-central-1'
    ]
]);

// Creating an Amazon DynamoDb client will use the "eu-central-1" AWS Region
$client = $sdk->createDynamoDb();
```

サービス固有の設定値は、サービス固有の値とルートレベルの値の共用体です (つまり、サービス固有の値がルートレベルの値にシャローマージされている)。

> **Note**
>
> アプリケーションでクライアントの複数のインスタンスを使用している場合は、Sdk クラスを使用してクライアントを作成することを強くお勧めします。Sdk クラスでは、各 SDK クライアントに対して同じ HTTP クライアントが自動的に使用されるため、異なるサービス用の SDK クライアントでノンブロッキング HTTP リクエストを実行できます。SDK クライアントで同じ HTTP クライアントを使用していない場合は、SDK が送信する HTTP リクエストによってサービス間でのプロミスオーケストレーションがブロックされます。

サービスオペレーションの実行

サービスオペレーションは、クライアントオブジェクトにある同じ名前のメソッドを呼び出すことによって実行できます。たとえば、Amazon S3 の PutObject オペレーションを実行するには、`Aws\S3\S3Client::putObject()` メソッドを呼び出します。

インポート

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
```

サンプルコード

```
// Use the us-east-2 region and latest version of each client.
$sharedConfig = [
    'profile' => 'default',
    'region' => 'us-east-2'
];

// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk($sharedConfig);

// Use an Aws\Sdk class to create the S3Client object.
$s3Client = $sdk->createS3();

// Send a PutObject request and get the result object.
$result = $s3Client->putObject([
    'Bucket' => 'my-bucket',
    'Key' => 'my-key',
    'Body' => 'this is the body!'
]);

// Download the contents of the object.
$result = $s3Client->getObject([
    'Bucket' => 'my-bucket',
    'Key' => 'my-key'
]);

// Print the body of the result by indexing into the result object.
echo $result['Body'];
```

クライアントで使用可能なオペレーションおよび入力と出力の構造体は、サービス記述ファイルに基づいて実行時に定義されます。クライアントの作成時に、バージョン (例: "2006-03-01"、"latest") を指定する必要があります。SDK は、提供されたバージョンに基づいて、対応する設定ファイルを見つけます。

putObject() などのオペレーションメソッドはすべて、オペレーションのパラメーターを表す連想配列である単一の引数を受け入れます。この配列の構造体 (および結果オブジェクトの構造体) は、SDK の API ドキュメント (たとえば、putObject オペレーションの API ドキュメントを参照) で各オペレーションに対して定義されています。

HTTP ハンドラーオプション

特別な @http パラメーターを使用することによって、基になるハンドラーでリクエストがどのように実行されるかを調整することもできます。@http パラメータに含めることができるオプションは、「http」クライアントオプションを使用してそのクライアントをインスタンス化するときに設定できるオプションと同じです。

```
// Send the request through a proxy
$result = $s3Client->putObject([
    'Bucket' => 'my-bucket',
    'Key'    => 'my-key',
    'Body'   => 'this is the body!',
    '@http'  => [
        'proxy' => 'http://192.168.16.1:10'
    ]
]);
```

非同期リクエスト

SDK の非同期機能を使用して、複数のコマンドを同時に送信できます。オペレーション名の後に Async を付けることによって、リクエストを非同期的に送信できます。そうすると、そのリクエストが開始されて promise が返されます。その promise には、成功すると結果オブジェクトが満たされ、失敗すると例外で拒否されます。これにより、複数の promise を作成して、基になる HTTP ハンドラーがリクエストを転送するときに複数の HTTP リクエストを同時に送信できるようになります。

インポート

```
require 'vendor/autoload.php';
use Aws\S3\S3Client;
use Aws\Exception\AwsException;
```

サンプルコード

```
// Create an SDK class used to share configuration across clients.
```

```
$sdk = new Aws\Sdk([
    'region'   => 'us-west-2'
]);
// Use an Aws\Sdk class to create the S3Client object.
$s3Client = $sdk->createS3();
//Listing all S3 Bucket
$CompleteSynchronously = $s3Client->listBucketsAsync();
// Block until the result is ready.
$CompleteSynchronously = $CompleteSynchronously->wait();
```

promise の `wait` メソッドを使用すると、promise が同期的に完了するように強制できます。promise が同期的に完了するように強制すると、デフォルトでは、その promise の状態が「ラップ解除」されます。つまり、promise の結果が返されるか、または検出された例外がスローされます。promise の `wait()` を呼び出した場合、そのプロセスは HTTP リクエストが完了するまでブロックされ、結果が入力されるかまたは例外がスローされます。

イベントループライブラリで SDK を使用する場合は、結果はブロックしないでください。代わりに、オペレーションが完了したときに、結果の `then()` メソッドを使用して、解決または拒否された promise にアクセスします。

インポート

```
require 'vendor/autoload.php';
use Aws\S3\S3Client;
use Aws\Exception\AwsException;
```

サンプルコード

```
// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk([
    'region'   => 'us-west-2'
]);
// Use an Aws\Sdk class to create the S3Client object.
$s3Client = $sdk->createS3();
```

```
$promise = $s3Client->listBucketsAsync();
$promise
    ->then(function ($result) {
        echo 'Got a result: ' . var_export($result, true);
    })
    ->otherwise(function ($reason) {
```

```
        echo 'Encountered an error: ' . $reason->getMessage();
    });
```

結果オブジェクトの使用

オペレーションの実行が正常に終了すると、Aws\Result オブジェクトが返されます。SDK では、サービスの RAW XML データまたは RAW JSON データを返すのではなく、レスポンスデータが連想配列の構造体に配置されます。特定のサービスおよび基になるレスポンス構造体に関する知識に基づいて、データの一部の側面が正規化されます。

PHP 連想配列などの AWSResult オブジェクトからデータにアクセスできます。

インポート

```
require 'vendor/autoload.php';
use Aws\S3\S3Client;
use Aws\Exception\AwsException;
```

サンプルコード

```
// Use the us-east-2 region and latest version of each client.
$sharedConfig = [
    'profile' => 'default',
    'region' => 'us-east-2',
];

// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk($sharedConfig);

// Use an Aws\Sdk class to create the S3Client object.
$s3 = $sdk->createS3();
$result = $s3->listBuckets();
foreach ($result['Buckets'] as $bucket) {
    echo $bucket['Name'] . "\n";
}

// Convert the result object to a PHP array
$array = $result->toArray();
```

結果オブジェクトの内容は、実行されたオペレーションやサービスのバージョンによって異なります。各 API オペレーションの結果の構造体は、各オペレーションの API ドキュメントに記載されています。

SDK には、JSON データや PHP 配列の検索および操作に使用される DSL である JMESPath が組み込まれています。結果オブジェクトには、より詳細な宣言によって結果からデータを抽出できる search() メソッドが含まれています。

サンプルコード

```
$s3 = $sdk->createS3();
$result = $s3->listBuckets();
```

```
$names = $result->search('Buckets[].Name');
```

エラー処理

同期エラー処理

オペレーションの実行中にエラーが発生すると、例外がスローされます。そのため、コード内でエラーを処理する必要がある場合は、オペレーションを try/catch ブロック内に配置します。SDK では、エラーが発生するとサービス固有の例外がスローされます。

次の例では Aws\S3\S3Client を使用しています。エラーがある場合にスローされる例外の型は Aws\S3\Exception\S3Exception です。SDK によってスローされるサービス固有のすべての例外は Aws\Exception\AwsException クラスから拡張されます。このクラスには、リクエスト ID、エラーコード、エラータイプなどの、失敗に関する有益な情報が含まれています。これをサポートする一部のサービスでは、レスポンスデータが連想配列構造 (Aws\Result オブジェクトに類似) に強制変換されることに注意してください。この構造は、通常の PHP 連想配列のようにアクセスできます。この toArray() メソッドは、そのようなデータを返します (存在する場合)。

インポート

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;
use Aws\S3\Exception\S3Exception;
```

サンプルコード

```
// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk([
```

```
    'region'   => 'us-west-2'
]);

// Use an Aws\Sdk class to create the S3Client object.
$s3Client = $sdk->createS3();

try {
    $s3Client->createBucket(['Bucket' => 'my-bucket']);
} catch (S3Exception $e) {
    // Catch an S3 specific exception.
    echo $e->getMessage();
} catch (AwsException $e) {
    // This catches the more generic AwsException. You can grab information
    // from the exception using methods of the exception object.
    echo $e->getAwsRequestId() . "\n";
    echo $e->getAwsErrorType() . "\n";
    echo $e->getAwsErrorCode() . "\n";

    // This dumps any modeled response data, if supported by the service
    // Specific members can be accessed directly (e.g. $e['MemberName'])
    var_dump($e->toArray());
}
```

非同期エラー処理

非同期リクエストを送信した場合には例外はスローされません。代わりに、返された promise の `then()` または `otherwise()` メソッドを使用して、結果またはエラーを受け取る必要があります。

インポート

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;
use Aws\S3\Exception\S3Exception;
```

サンプルコード

```
//Asynchronous Error Handling
```

```
$promise = $s3Client->createBucketAsync(['Bucket' => 'my-bucket']);
$promise->otherwise(function ($reason) {
    var_dump($reason);
});

// This does the same thing as the "otherwise" function.
$promise->then(null, function ($reason) {
    var_dump($reason);
});
```

promise をラップ解除して、例外がスローされるようにできます。

インポート

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;
use Aws\S3\Exception\S3Exception;
```

サンプルコード

```
$promise = $s3Client->createBucketAsync(['Bucket' => 'my-bucket']);
```

```
//throw exception
try {
    $result = $promise->wait();
} catch (S3Exception $e) {
    echo $e->getMessage();
}
```

AWS SDK for PHP バージョン 3 の設定

クライアントコンストラクタは、クライアントコンストラクタまたは `Aws\Sdk` クラスで指定できます。特定のタイプのクライアントに指定されるオプションの配列は、作成するクライアントによって異なります。これらのカスタムクライアント設定オプションについては、各クライアントの API ドキュメントで説明されています。

一部の設定オプションは、環境変数または AWS 設定ファイルに基づいてデフォルト値をチェックして使用することに注意してください。デフォルトでは、チェックされる設定ファイルは、ホームディレクトリの `.aws/config`で、通常、`~/.aws/config` です。ただし、環境変数 `AWS_CONFIG_FILE` を使用して、デフォルトの設定ファイルの場所を設定できます。例えば、これは `open_basedir` などで特定のディレクトリへのファイルアクセスを制限する場合に便利です。

共有 AWS `config` ファイルと `credentials` ファイルの場所と形式について詳しくは、AWSSDK およびツールリファレンスガイドの「設定」を参照してください。

AWS 構成ファイルまたは環境変数として設定できるすべてのグローバル構成設定について詳しくは、AWS SDK およびツールリファレンスガイドの「構成と認証設定のリファレンス」を参照してください。

設定オプション

- api_provider
- 認証情報
- デバッグ
- stats
- エンドポイント
- endpoint_provider
- endpoint_discovery
- handler
- http
- http_handler
- profile
- region
- retries
- scheme
- service
- signature_provider
- signature_version
- ua_append
- use_aws_shared_config_files
- validate

- version

以下の例では、Amazon S3 のクライアントコンストラクタにオプションを渡す方法を示しています。

```
use Aws\S3\S3Client;

$options = [
    'region'            => 'us-west-2',
    'version'           => '2006-03-01',
    'signature_version' => 'v4'
];

$s3Client = new S3Client($options);
```

クライアントの作成の詳細については、「基本使用法ガイド」を参照してください。

api_provider

タイプ

```
callable
```

タイプ、サービス、およびバージョン引数を受け入れて、対応する設定データの配列を返す、PHP の呼び出し可能関数。タイプの値は `api`、`waiter`、`paginator` のいずれかです。

デフォルトでは、SDK は、SDK の `Aws\Api\FileSystemApiProvider` フォルダから API ファイルをロードする `src/data` のインスタンスを使用します。

認証情報

タイプ

```
array|Aws\CacheInterface|Aws\Credentials\CredentialsInterface|bool|
callable
```

特定の認証情報インスタンスを使用するための `Aws\Credentials\CredentialsInterface` オブジェクトを渡します。以下では、IAM Identity Center 認証情報プロバイダを使用することを指定しています。このプロバイダは SSO 認証情報プロバイダとも呼ばれます。

```
$credentials = Aws\Credentials\CredentialProvider::sso('profile default');

$s3 = new Aws\S3\S3Client([
    'region'      => 'us-west-2',
    'credentials' => $credentials
]);
```

名前付きプロファイルを使用する場合は、前の例の「default」をプロファイル名に置き換えてください。名前付きプロファイルの設定について詳しくは、AWS SDK およびツールリファレンスガイドの「共有 config と credentials ファイル」を参照してください。

使用する認証情報プロバイダを指定せず、認証情報プロバイダチェーンに依存している場合、認証失敗によるエラーメッセージは通常一般的なものです。有効な認証情報を確認しているソースのリストの最後のプロバイダから生成されますが、そのプロバイダは使用しようとしているプロバイダではない可能性があります。どの認証情報プロバイダを使用するかを指定すると、表示されるエラーメッセージはそのプロバイダのみから出力されるため、より有用で関連性の高いものになります。認証情報をチェックするソースチェーンの詳細については、AWS SDK およびツールリファレンスガイドの「認証情報プロバイダチェーン」を参照してください。

Null 認証情報を使用し、リクエストに署名しないように、false を渡します。

```
$s3 = new Aws\S3\S3Client([
    'region'      => 'us-west-2',
    'credentials' => false
]);
```

関数を使用して認証情報を作成するために、呼び出し可能な認証情報プロバイダ関数を渡します。

```
use Aws\Credentials\CredentialProvider;

// Only load credentials from environment variables
$provider = CredentialProvider::env();

$s3 = new Aws\S3\S3Client([
    'region'      => 'us-west-2',
    'credentials' => $provider
]);
```

複数のプロセスにわたってデフォルトプロバイダーチェーンで返される値をキャッシュするために、Aws\CacheInterface のインスタンスにキャッシュされた認証情報を渡します。

```
use Aws\Credentials\CredentialProvider;
use Aws\PsrCacheAdapter;
use Symfony\Component\Cache\Adapter\FilesystemAdapter;


$cache = new PsrCacheAdapter(new FilesystemAdapter);
$provider = CredentialProvider::defaultProvider();
$cachedProvider = CredentialProvider::cache($provider, $cache);

$s3 = new Aws\S3\S3Client([
    'region' => 'us-west-2',
    'credentials' => $cachedProvider
]);
```

クライアントへの認証情報の提供の詳細については、「AWS SDK for PHP バージョン 3 の認証情報」を参照してください。

> **Note**
>
> 認証情報のロードおよび検証は、それが使用されるときまで遅延されます。

デバッグ

タイプ

```
bool|array
```

各転送に関するデバッグ情報を出力します。デバッグ情報には、トランザクションが準備されネットワーク経由で送信されるときの、トランザクションの状態の変更に関する情報が含まれています。また、デバッグ出力には、クライアントで使用される特定の HTTP ハンドラー (例: cURL のデバッグ出力) に関する情報も含まれています。

`true` に設定すると、リクエストの送信時にデバッグ情報が表示されます。

```
$s3 = new Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'debug'   => true
]);
```

```
// Perform an operation to see the debug output
$s3->listBuckets();
```

または、以下のキーを持つ連想配列を指定できます。

logfn (callable)

ログメッセージで呼び出される関数。デフォルトでは、PHP の echo 関数が使用されます。

stream_size (int)

ストリームのサイズがこの数より大きい場合、ストリームデータは記録されません。0 に設定すると、ストリームデータは一切記録されません。

scrub_auth (bool)

false に設定すると、記録されたメッセージでの認証データのスクラブが無効になります (つまり、AWS アクセスキー ID や署名が logfn に渡されます)。

http (bool)

false に設定すると、低レベルの HTTP ハンドラー (例: cURL の詳細出力) の「debug」機能が無効になります。

auth_headers (array)

置き換え先の値にマッピングされている、置き換えるヘッダーの、キーと値のマッピングに設定します。scrub_auth が true に設定されていない場合、これらの値は使用されません。

auth_strings (array)

置き換え先にマッピングするための正規表現の、キーと値のマッピングに設定します。scrub_auth が true に設定されている場合に、これらの値は認証データスクラバーで使用されます。

```
$s3 = new Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'debug'   => [
        'logfn'        => function ($msg) { echo $msg . "\n"; },
        'stream_size'  => 0,
        'scrub_auth'   => true,
        'http'         => true,
        'auth_headers' => [
```

```
            'X-My-Secret-Header' => '[REDACTED]',
        ],
        'auth_strings' => [
            '/SuperSecret=[A-Za-z0-9]{20}/i' => 'SuperSecret=[REDACTED]',
        ],
    ]
]);

// Perform an operation to see the debug output
$s3->listBuckets();
```

> **Note**
>
> このオプションは、http デバッグオプションによって生成された基礎となる HTTP ハンドラー情報も出力します。デバッグ出力は、AWS SDK for PHP での問題を診断する場合に非常に役立ちます。SDK に関する問題をオープンする場合は、分離した障害ケースのデバッグ出力を提供してください。

stats

タイプ

```
bool|array
```

転送の統計情報を、SDK オペレーションによって返されるエラーと結果にバインドします。

`true` に設定すると、送信されたリクエストに関する統計情報が収集されます。

```
$s3 = new Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'stats'   => true
]);

// Perform an operation
$result = $s3->listBuckets();
// Inspect the stats
$stats = $result['@metadata']['transferStats'];
```

または、以下のキーを持つ連想配列を指定できます。

retries (bool)

`true` に設定すると、再試行に関するレポートが有効になります。デフォルトでは、再試行の統計情報が収集されて、返されます。

http (bool)

に設定するtrueと、下位レベルの HTTP アダプターからの統計の収集が有効になります (例: で返される値 GuzzleHttpTransferStats)。このオプションの効果が生じるには、HTTP ハンドラーで __on_transfer_stats オプションがサポートされている必要があります。HTTP 統計情報は連想配列のインデックス付き配列として返されます。それぞれの連想配列には、1 つのリクエストに対して、クライアントの HTTP ハンドラーによって返される転送統計情報が含まれています。デフォルトでは無効になっています。

リクエストが再試行された場合、各リクエストの転送統計情報が返され、最初のリクエストの統計情報が `$result['@metadata']['transferStats']['http'][0]` に、2 回目のリクエストの統計情報が `$result['@metadata']['transferStats']['http'][1]` に入っています (以下同様)。

timer (bool)

`true` に設定すると、1 つのオペレーションの合計所要時間 (秒単位) をレポートするコマンドタイマーが有効になります。デフォルトでは無効になっています。

```
$s3 = new Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'stats'   => [
        'retries'      => true,
        'timer'        => false,
        'http'         => true,
    ]
]);

// Perform an operation
$result = $s3->listBuckets();
// Inspect the HTTP transfer stats
$stats = $result['@metadata']['transferStats']['http'];
// Inspect the number of retries attempted
$stats = $result['@metadata']['transferStats']['retries_attempted'];
// Inspect the total backoff delay inserted between retries
$stats = $result['@metadata']['transferStats']['total_retry_delay'];
```

エンドポイント

タイプ

```
string
```

ウェブサービスの完全な URI。これは、アカウント固有のエンドポイントを使用する AWS Elemental MediaConvert などのサービスに必須です。これらのサービスについては、`describeEndpoints` メソッドを使用してこのエンドポイントをリクエストしてください。

これは、カスタムエンドポイント (例: Amazon S3 または Amazon DynamoDB Local のローカルバージョン) に接続する場合にのみ必要です。

Amazon DynamoDB ローカルへの接続の例を次に示します。

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'version'  => '2012-08-10',
    'region'   => 'us-east-1',
    'endpoint' => 'http://localhost:8000'
]);
```

使用可能な AWS リージョンとエンドポイントの一覧については、「AWS リージョンとエンドポイント」を参照してください。

endpoint_provider

タイプ

```
Aws\EndpointV2\EndpointProviderV2|callable
```

「service」キーや「region」キーなど、オプションのハッシュを受け入れる EndpointProviderV2 または PHP 呼び出し可能なオプションインスタンス。NULL またはエンドポイントデータのハッシュ (「エンドポイント」キーが必要) を返します。

最小限のエンドポイントプロバイダーを作成する方法の例を次に示します。

```
$provider = function (array $params) {
    if ($params['service'] == 'foo') {
        return ['endpoint' => $params['region'] . '.example.com'];
```

```
    }
    // Return null when the provider cannot handle the parameters
    return null;
});
```

endpoint_discovery

タイプ

```
array|Aws\CacheInterface|Aws\EndpointDiscovery\ConfigurationInterface|
callable
```

エンドポイント検出では、エンドポイント検出をサポートするサービス API に対する正しいエンドポイントを識別して接続します。エンドポイント検出をサポートするが、これを必要としないサービスの場合、クライアントの作成中に endpoint_discovery を有効にします。サービスがエンドポイント検出をサポートしていない場合、この設定は無視されます。

Aws\EndpointDiscovery\ConfigurationInterface

サービスが指定するオペレーションのサービス API の適切なエンドポイントへの自動接続を可能にするオプションの設定プロバイダー。

Aws\EndpointDiscovery\Configuration オブジェクトでは 2 つのオプションを使用できます。1 つはエンドポイント検出が有効な場合にそれを示すブール値「enabled」で、もう 1 つはエンドポイントキャッシュのキーの最大数を示す整数「cache_limit」です。

作成されるクライアントごとに、エンドポイント検出の特定の設定を使用するために Aws\EndpointDiscovery\Configuration オブジェクトを渡します。

```
use Aws\EndpointDiscovery\Configuration;
use Aws\S3\S3Client;

$enabled = true;
$cache_limit = 1000;

$config = new Aws\EndpointDiscovery\Configuration (
    $enabled,
    $cache_limit
);

$s3 = new Aws\S3\S3Client([
```

```
    'region' => 'us-east-2',
    'endpoint_discovery' => $config,

]);
```

複数のプロセスにわたってエンドポイント検出で返される値をキャッシュするために、Aws\CacheInterface のインスタンスを渡します。

```
use Aws\DoctrineCacheAdapter;
use Aws\S3\S3Client;
use Doctrine\Common\Cache\ApcuCache;

$s3 = new S3Client([
    'region'      => 'us-west-2',
    'endpoint_discovery' => new DoctrineCacheAdapter(new ApcuCache),
]);
```

エンドポイント検出に配列を渡します。

```
use Aws\S3\S3Client;

$s3 = new S3Client([
    'region'      => 'us-west-2',
    'endpoint_discovery' => [
        'enabled' => true,
        'cache_limit' => 1000
    ],
]);
```

handler

タイプ

```
  callable
```

コマンドオブジェクトとリクエストオブジェクトを受け入れて、GuzzleHttp\Promise\PromiseInterface オブジェクトで満たすかまたは Aws\ResultInterface で拒否する promise (Aws\Exception\AwsException) を返すハンドラーです。ハンドラーはターミナルであり、コマンドを満たすことになっているため、ハンドラーは next ハンドラーを受け入れません。ハンドラーが指定されていない場合は、デフォルトの Guzzle ハンドラーが使用されます。

Aws\MockHandler を使用して、模擬結果を返すか、または Mock 例外をスローできます。結果または例外をキューに入れると、 MockHandler はそれらを FIFO 順序でキューに入れます。

```
use Aws\Result;
use Aws\MockHandler;
use Aws\DynamoDb\DynamoDbClient;
use Aws\CommandInterface;
use Psr\Http\Message\RequestInterface;
use Aws\Exception\AwsException;

$mock = new MockHandler();

// Return a mocked result
$mock->append(new Result(['foo' => 'bar']));

// You can provide a function to invoke; here we throw a mock exception
$mock->append(function (CommandInterface $cmd, RequestInterface $req) {
    return new AwsException('Mock exception', $cmd);
});

// Create a client with the mock handler
$client = new DynamoDbClient([
    'region' => 'us-east-1',
    'handler' => $mock
]);

// Result object response will contain ['foo' => 'bar']
$result = $client->listTables();

// This will throw the exception that was enqueued
$client->listTables();
```

http

タイプ

```
array
```

SDK によって作成される HTTP リクエストと転送に適用する HTTP オプションの配列に設定します。

この SDK では以下の設定オプションがサポートされています。

cert

タイプ

```
string|array
```

PEM 形式のクライアント側の証明書を指定します。

- 証明書ファイルのみへのパスの文字列として設定します。

```
use Aws\S3\S3Client;

$client = new S3Client([
    'region'  => 'us-west-2',
    'http'    => ['cert' => '/path/to/cert.pem']
]);
```

- パスとパスワードを含む配列として設定します。

```
use Aws\S3\S3Client;

$client = new S3Client([
    'region'  => 'us-west-2',
    'http'    => [
        'cert' => ['/path/to/cert.pem', 'password']
    ]
]);
```

connect_timeout

サーバーへの接続の試行時に待機する秒数を記述する浮動小数点。`0` に設定すると、無期限に待機します (デフォルト動作)。

```
use Aws\DynamoDb\DynamoDbClient;

// Timeout after attempting to connect for 5 seconds
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
```

```
        'connect_timeout' => 5
    ]
]);
```

デバッグ

タイプ

```
bool|resource
```

デバッグ情報を出力するように、基になる HTTP ハンドラーに指示します。HTTP ハンドラーから提供されるデバッグ情報は、HTTP ハンドラーごとに異なります。

- `true` を渡すと、デバッグ出力は STDOUT に書き込まれます。
- `resource` で返された `fopen` を渡すと、デバッグ出力は特定の PHP ストリームリソースに書き込まれます。

decode_content

タイプ

```
bool
```

圧縮されたレスポンスの本文を解凍するように、基になる HTTP ハンドラーに指示します。有効になっていない場合、圧縮されたレスポンス本文は `GuzzleHttp\Psr7\InflateStream` を使用して解凍されます。

> **Note**
>
> SDK のデフォルト HTTP ハンドラーでは、コンテンツのデコードはデフォルトで有効になっています。下位互換性のために、このデフォルトは変更できません。圧縮されたファイルを Amazon S3 に保存する場合は、S3 クライアントレベルでコンテンツのデコードを無効にすることをお勧めします。

```
use Aws\S3\S3Client;
use GuzzleHttp\Psr7\InflateStream;

$client = new S3Client([
```

```
    'region'  => 'us-west-2',
    'http'    => ['decode_content' => false],
]);

$result = $client->getObject([
    'Bucket' => 'my-bucket',
    'Key'    => 'massize_gzipped_file.tgz'
]);

$compressedBody = $result['Body']; // This content is still gzipped
$inflatedBody = new InflateStream($result['Body']); // This is now readable
```

delay

タイプ

```
int
```

リクエストを送信する前の遅延の時間数 (ミリ秒単位)。これは、リクエストを再試行する前に遅延するためによく使用されます。

expect

タイプ

```
bool|string
```

このオプションは、基盤となる HTTP ハンドラーに渡されます。デフォルトでは、リクエストの本文が 1 MB を超えたときに Expect: 100-Continue ヘッダーが設定されます。`true` または `false` により、すべてのリクエストでヘッダーが有効または無効になります。整数を使用する場合、この設定を超える本文を持つリクエストのみが、ヘッダーを使用します。整数として使用した場合、本文のサイズが不明なときは Expect ヘッダーが送信されます。

> **Warning**
>
> Expect ヘッダーを無効にすると、サービスは認証エラーやその他のエラーを返すことができなくなる場合があります。このオプションの設定には注意が必要です。

progress

タイプ

```
callable
```

転送の進行状況が作成されたときに呼び出す関数を定義します。この関数は以下の次の引数を受け入れます。

1. ダウンロードすることになっている総バイト数。
2. これまでにダウンロード済みのバイト数。
3. アップロードすることになっている総バイト数。
4. これまでにアップロード済みのバイト数。

```
use Aws\S3\S3Client;

$client = new S3Client([
    'region'  => 'us-west-2'
]);

// Apply the http option to a specific command using the "@http"
// command parameter
$result = $client->getObject([
    'Bucket' => 'my-bucket',
    'Key'    => 'large.mov',
    '@http' => [
        'progress' => function ($expectedDl, $dl, $expectedUl, $ul) {
            printf(
                "%s of %s downloaded, %s of %s uploaded.\n",
                $expectedDl,
                $dl,
                $expectedUl,
                $ul
            );
        }
    ]
]);
```

proxy

タイプ

```
string|array
```

proxy オプションを使用することによって、プロキシを介して AWS のサービスと接続できます。

- すべてのタイプの URI に対するプロキシに接続するための文字列値を指定します。このプロキシ文字列値には、スキーム、ユーザー名、およびパスワードを含めることができます。例えば、「"http://username:password@192.168.16.1:10"」と入力します。
- プロキシ設定の連想配列を指定します。キーは URI のスキームであり、値は指定の URI に対するプロキシです (つまり、「http」と「https」のエンドポイントに対して異なるプロキシを指定できます)。

```
use Aws\DynamoDb\DynamoDbClient;

// Send requests through a single proxy
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
        'proxy' => 'http://192.168.16.1:10'
    ]
]);

// Send requests through a different proxy per scheme
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
        'proxy' => [
            'http' => 'tcp://192.168.16.1:10',
            'https' => 'tcp://192.168.16.1:11',
        ]
    ]
]);
```

HTTP_PROXY 環境変数を使用して「http」プロトコル固有のプロキシを、HTTPS_PROXY 環境変数を使用して「https」固有のプロキシをそれぞれ設定できます。

sink

タイプ

```
resource|string|Psr\Http\Message\StreamInterface
```

`sink` オプションは、オペレーションのレスポンスデータのダウンロード先を制御します。

- `resource` で返される `fopen` を指定すると、レスポンス本文は PHP ストリームにダウンロードされます。
- ディスク上のファイルのパスを `string` 値として指定すると、レスポンス本文はディスク上の特定のファイルにダウンロードされます。
- `Psr\Http\Message\StreamInterface` を指定すると、レスポンス本文は特定の PSR ストリームオブジェクトにダウンロードされます。

> **Note**
>
> SDK はデフォルトでは、レスポンス本文を PHP の一時ストリームにダウンロードします。つまり、本文のサイズが 2 MB に達するまでデータはメモリに保持され、2 MB に達した時点でそのデータはディスク上の一時ファイルに書き込まれます。

synchronous

タイプ

```
bool
```

`synchronous` オプションを true に設定すると、結果をブロックする予定であることが、基になる HTTP ハンドラーに通知されます。

ストリーム

タイプ

```
bool
```

true に設定すると、ウェブサービスからのレスポンスの本文を、最初にすべてダウンロードするのではなく、ストリーミングする予定であることが、基になる HTTP ハンドラーに通知されます。例えば、Amazon S3 のストリームラッパークラスでは、データが確実にストリーミングされるように、このオプションを使用します。

timeout

タイプ

```
float
```

リクエストのタイムアウトを秒数で記述する浮動小数点。0 に設定すると、無期限に待機します (デフォルト動作)。

```
use Aws\DynamoDb\DynamoDbClient;

// Timeout after 5 seconds
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
        'timeout' => 5
    ]
]);
```

認証

タイプ

```
bool|string
```

verifyhttp オプションを使用して、SDK でのピアの SSL/TLS 証明書検証動作をカスタマイズできます。

- true に設定すると、ピアの SSL/TLS 証明書検証が有効になり、オペレーティングシステムで提供されているデフォルトの CA バンドルが使用されます。
- false に設定すると、ピアの証明書検証が無効になります (これは安全ではありません)。
- CA 証明書バンドルのパスを指定する文字列に設定すると、カスタム CA バンドルを使用した検証が有効になります。

指定した CA バンドルがシステムに見つからずにエラーを受け取った場合は、SDK に対して CA バンドルのパスを指定します。特定の CA バンドルを使用する必要がない場合は、ここからダウンロードできる、一般的に使用される CA バンドル (cURL のメンテナンス担当者によって維持されている) が Mozilla によって提供されています。使用可能な CA バンドルをディスク上に準備したら、PHP の .ini 設定 `openssl.cafile` でそのファイルのパスを指すように設定できます。そうすると、`verify` リクエストオプションを省略できます。SSL 証明書の詳細については、cURL ウェブサイトを参照してください。

```
use Aws\DynamoDb\DynamoDbClient;

// Use a custom CA bundle
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
        'verify' => '/path/to/my/cert.pem'
    ]
]);

// Disable SSL/TLS verification
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
        'verify' => false
    ]
]);
```

http_handler

タイプ

```
callable
```

`http_handler` オプションは、SDK を他の HTTP クライアントと統合するために使用されます。`http_handler` オプションは、`Psr\Http\Message\RequestInterface` オブジェクトと、コマンドに適用する `http` オプションの配列を受け入れて、`GuzzleHttp\Promise\PromiseInterface` オブジェクトで満たすかまたは以下の例外データで拒否する `Psr\Http\Message\ResponseInterface` オブジェクトを返す関数です。

- `exception` - (`\Exception`) 発生した例外。
- `response` - (`Psr\Http\Message\ResponseInterface`) 受信したレスポンス (存在する場合)。

- `connection_error` - (bool) `true` に設定すると、そのエラーが接続エラーとしてマークされます。また、この値を `true` 設定すると、SDK が必要に応じてオペレーションを自動的に再試行できるようになります。

SDK は、指定された `http_handler` を `handler` オブジェクトでラップすることによって、指定された `http_handler` を通常の `Aws\WrappedHttpHandler` オプションに自動的に変換します。

デフォルトでは、SDK は Guzzle を HTTP ハンドラーとして使用します。ここで別の HTTP ハンドラーを指定するか、Guzzle クライアントに独自のカスタム定義オプションを提供することができます。

TLS バージョンの設定

1 つのユースケースは、Curl が環境にインストールされていることを前提に、Curl で Guzzle が使用する TLS バージョンを設定することです。サポートされている TLS のバージョンに関する Curl バージョンの制約に注意してください。デフォルトでは、最新バージョンが使用されます。TLS バージョンが明示的に設定されていて、リモートサーバーがこのバージョンをサポートしていない場合、以前の TLS バージョンを使用する代わりにエラーが発生します。

debug クライアントオプションを true に設定し、SSL 接続の出力を確認することによって、特定のクライアント操作に使用されている TLS のバージョンを決定できます。その行は次のようになります。`SSL connection using TLSv1.2`

Guzzle 6 で TLS 1.2を設定する例:

```
use Aws\DynamoDb\DynamoDbClient;
use Aws\Handler\GuzzleV6\GuzzleHandler;
use GuzzleHttp\Client;

$handler = new GuzzleHandler(
    new Client([
        'curl' => [
            CURLOPT_SSLVERSION => CURL_SSLVERSION_TLSv1_2
        ]
    ])
);

$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http_handler' => $handler
]);
```

> **Note**
>
> `http_handler` オプションは、指定されたどの `handler` オプションよりも優先されます。

profile

タイプ

`string`

「profile」オプションは、認証情報がホームディレクトリ (通常は AWS) にある `~/.aws/credentials` 認証情報ファイルから作成されるときに使用されるプロファイルを指定します。この設定は `AWS_PROFILE` 環境変数をオーバーライドします。

> **Note**
>
> 「profile」オプションを指定すると、`"credentials"` オプションは無視され、設定ファイル内の認証情報関連の設定 AWS (通常`~/.aws/config`) は無視されます。

```
// Use the "production" profile from your credentials file
$ec2 = new Aws\Ec2\Ec2Client([
    'version' => '2014-10-01',
    'region'  => 'us-west-2',
    'profile' => 'production'
]);
```

認証情報の設定および .ini ファイル形式の詳細については、「AWS SDK for PHP バージョン 3 の認証情報」を参照してください。

region

タイプ

`string`

必須

true

接続先の AWS リージョン。利用可能なリージョンのリストについては、「AWS のリージョンとエンドポイント」を参照してください。

```
// Set the Region to the EU (Frankfurt) Region
$s3 = new Aws\S3\S3Client([
    'region'  => 'eu-central-1',
    'version' => '2006-03-01'
]);
```

retries

タイプ

```
int|array|Aws\CacheInterface|Aws\Retry\ConfigurationInterface|callable
```

デフォルト値

```
int(3)
```

クライアントに対して再試行モードと許可される最大再試行回数を設定します。`0` を渡すと、再試行が無効になります。

次の 3 つの再試行モードがあります。

- `legacy` - デフォルトのレガシー再試行実装
- `standard` - 成功しそうもない再試行を防ぐため、再試行クォータシステムを追加
- `adaptive` - 標準モードに基づいて構築され、クライアント側のレートリミッターを追加します。このモードは実験的と見なされることに注意してください。

再試行の設定は、モードと各リクエストに使用される最大試行回数で構成されます。設定は、次の優先順位に従って、いくつかの異なる場所で設定できます。

優先順位

再試行設定の優先順位は次のとおりです (1 は 2-3 を上書きします)。

1. クライアント設定オプション
2. 環境変数
3. AWS 共有 Config ファイル

環境変数

- AWS_RETRY_MODE を legacy、standard、または adaptive に設定します。
- AWS_MAX_ATTEMPTS - リクエストあたりの最大試行回数の整数値に設定

共有 Config ファイルキー

- retry_mode を legacy、standard、または adaptive に設定します。
- max_attempts - リクエストあたりの最大試行回数の整数値に設定

クライアント設定

次の例では、Amazon DynamoDB クライアントの再試行を無効にしています。

```
// Disable retries by setting "retries" to 0
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region'  => 'us-west-2',
    'retries' => 0
]);
```

次の例では、整数を渡します。デフォルトでは、再試行回数が渡された状態で legacy モードになります。

```
// Disable retries by setting "retries" to 0
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region'  => 'us-west-2',
    'retries' => 6
]);
```

Aws\Retry\Configuration オブジェクトは、2 つのパラメータ、

再試行モードとリクエストごとの最大試行回数の整数を受け付けます。この例では、

再試行設定の Aws\Retry\Configuration オブジェクトを渡します。

```
use Aws\EndpointDiscovery\Configuration;
```

```
use Aws\S3\S3Client;

$enabled = true;
$cache_limit = 1000;

$config = new Aws\Retry\Configuration('adaptive', 10);

$s3 = new Aws\S3\S3Client([
    'region' => 'us-east-2',
    'retries' => $config,
]);
```

この例では、再試行設定のために配列を渡します。

```
use Aws\S3\S3Client;

$s3 = new S3Client([
    'region' => 'us-west-2',
    'retries' => [
        'mode' => 'standard',
        'max_attempts' => 7
    ],
]);
```

この例では、Aws\CacheInterface のインスタンスを渡して、デフォルトの再試行設定プロバイダーから返された値をキャッシュします。

```
use Aws\DoctrineCacheAdapter;
use Aws\S3\S3Client;
use Doctrine\Common\Cache\ApcuCache;

$s3 = new S3Client([
    'region' => 'us-west-2',
    'endpoint_discovery' => new DoctrineCacheAdapter(new ApcuCache),
]);
```

scheme

タイプ

string

デフォルト値

```
string(5) "https"
```

接続するときに使用される URI スキーム。SDK では、デフォルトでは「https」エンドポイント (つまり、SSL/TLS 接続) が使用されます。scheme を「http」に設定することによって、暗号化されていない「http」エンドポイントでのサービスへの接続を試行できます。

```
$s3 = new Aws\S3\S3Client([
    'version' => '2006-03-01',
    'region'  => 'us-west-2',
    'scheme'  => 'http'
]);
```

エンドポイントの一覧と http スキームをサポートするサービスについては、「AWS のリージョンとエンドポイント」を参照してください。

service

タイプ

```
string
```

必須

true

使用するサービスの名前。SDK で提供されているクライアントを使用する場合は、この値はデフォルトで指定されています (例: Aws\S3\S3Client)。このオプションは、SDK でまだ公開されていないがディスク上ですでに使用可能なサービスをテストするときに便利です。

signature_provider

タイプ

```
callable
```

署名バージョン名 (例: v4)、サービス名、および AWS リージョンを受け入れて、Aws\Signature\SignatureInterface オブジェクトを返すか、または指定されたパラメータの署名者をプロバイ

ダが作成できる場合は NULL を返す呼び出し可能関数。このプロバイダは、クライアントで使用される署名者を作成するために使用されます。

SDK の Aws\Signature\SignatureProvider クラスでは、カスタマイズされた署名プロバイダを作成するために使用できるさまざまな機能が提供されています。

signature_version

タイプ

```
string
```

サービスで使用されるカスタム署名バージョンを表す文字列 (例: v4)。オペレーション署名バージョンによって、このリクエストされた署名が必要に応じてオーバーライドされることがあります。

次の例は、署名バージョン 4 を使用するように Amazon S3 クライアントを設定する方法を示しています。

```
// Set a preferred signature version
$s3 = new Aws\S3\S3Client([
    'version'           => '2006-03-01',
    'region'            => 'us-west-2',
    'signature_version' => 'v4'
]);
```

> **Note**
>
> クライアントで使用する signature_provider は、指定した signature_version オプションを作成できる必要があります。SDK で使用されるデフォルトの signature_provider は、「v4」および「anonymous」署名バージョンの署名オブジェクトを作成できます。

ua_append

タイプ

```
string|string[]
```

デフォルト値

```
[]
```

HTTP ハンドラーに渡されるユーザーエージェント文字列に追加される、文字列または文字列の配列。

use_aws_shared_config_files

タイプ

```
bool|array
```

デフォルト値

```
bool(true)
```

'~/.aws/config 'および '~/.aws/credentials' の共有設定ファイルのチェックを無効にするには false に設定します。これは、AWS_CONFIG_FILE 環境変数をオーバーライドします。

validate

タイプ

```
bool|array
```

デフォルト値

```
bool(true)
```

false に設定すると、クライアント側のパラメーター検証が無効になります。検証をオフにすると、クライアントのパフォーマンスがわずかに向上することがありますが、その違いは無視できる程度です。

```
// Disable client-side validation
$s3 = new Aws\S3\S3Client([
    'version'  => '2006-03-01',
    'region'   => 'eu-west-1',
    'validate' => false
]);
```

検証オプションの連想配列に設定すると、以下の特定の検証の制約が有効になります。

- `required` - 必須パラメーターが存在することを検証します (デフォルトでオン)。
- `min` - 値の最小長さを検証します (デフォルトでオン)。
- `max` - 値の最大長を検証します。
- `pattern` - 値が正規表現と一致することを検証します。

```
// Validate only that required values are present
$s3 = new Aws\S3\S3Client([
    'version'  => '2006-03-01',
    'region'   => 'eu-west-1',
    'validate' => ['required' => true]
]);
```

version

タイプ

`string`

必須

false

このオプションは、使用するウェブサービスのバージョン (例: `2006-03-01`) を指定します。

SDK のバージョン 3.277.10 以降では、「version」オプションは不要になりました。「version」オプションを指定しない場合、SDK は、最新バージョンのサービスクライアントを使用します。

サービスクライアントを作成するときに「version」パラメータが必要なのは 2 つの状況です。

- 3.277.10 より前のバージョンの PHP SDK を使用する。
- バージョン 3.277.10 以降を使用していて、「latest」以外のバージョンをサービスクライアントに使用したいと思っている。

たとえば、次のスニペットでは SDK のバージョン 3.279.7 を使用していますが、`Ec2Client` の最新バージョンは使用していません。

```
$ec2Client = new \Aws\Ec2\Ec2Client([
    'version' => '2015-10-01',
    'region' => 'us-west-2'
]);
```

バージョン制約を指定すると、サービスに対して行われた互換性を破る変更の影響を受けないことが保証されます。

使用可能な API バージョンの一覧については、各クライアントの API ドキュメントのページを参照してください。特定の API バージョンをロードできない場合は、SDK のコピーの更新が必要になることがあります。

AWS SDK for PHP バージョン 3 の認証情報

AWS SDK で使用できる認証情報メカニズムの参照情報については、AWS SDK およびツールリファレンスガイドの 「認証情報とアクセス」を参照してください。

> **Important**
>
> セキュリティの観点から、AWS アクセスにはルートアカウントを使用しないことを強くお勧めします。最新のセキュリティ推奨事項については、必ず IAM ユーザーガイドの「IAM における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。

設定の優先順位

認証情報の引数を指定しないで新しいサービスクライアントを初期化すると、SDK はデフォルトの認証情報プロバイダチェーンを使用して AWS 認証情報を検索します。SDK では、チェーンのプロバイダの中で、最初にエラーのない認証情報を返したものを使用します。認証情報をチェックするソースチェーンの詳細については、AWS SDK およびツールリファレンスガイドの「認証情報プロバイダチェーン」を参照してください。

AWS SDK for PHP には、グローバル設定と認証情報プロバイダの値を検索するための一連のチェック項目があります。優先順位は次のとおりです。

1. コードまたはサービスクライアント自体に設定されている明示的な設定は、他の設定よりも優先されます。
2. 環境変数の認証情報を使用します。

Amazon EC2 インスタンス以外のマシンで開発作業を行う場合は、環境変数を設定すると便利です。

3. 共有 config および credentials ファイル.

これらは他の SDK や AWS CLI で使用されているのと同じファイルです。

認証情報プロバイダ

- 認証情報プロバイダの使用。

クライアントを構築するときに認証情報のカスタムロジックを提供します。

- IAM ロールを継承します。

IAM ロールは、インスタンス上のアプリケーションに対して、AWS の呼び出しを行うための一時的なセキュリティ認証情報を提供します。例えば、IAM ロールは複数の Amazon EC2 インスタンスに認証情報を分散して管理するための簡単な方法を提供します。

- AWS STS からの一時認証情報の使用

2 要素認証として多要素認証 (MFA) トークンを使用する場合は、AWS STS を使用して AWS のサービスにアクセスする一時的な認証情報をユーザーに提供するか、AWS SDK for PHP を使用します。

- 匿名クライアントの作成。

サービスで匿名アクセスが許可されている場合に、認証情報が関連付けられていないクライアントを作成します。

環境変数の認証情報を使用する

環境変数に認証情報を含めることで、AWS シークレットアクセスキーを誤って共有することを回避できます。本番稼働用ファイルで AWS アクセスキーを直接クライアントに追加することは絶対に避けてください。多くの開発者のアカウントがキーの漏洩によって侵害されています。

Amazon Web Services に対して認証する場合、SDK が最初に認証情報をチェックする場所はユーザーの環境変数です。SDK は `getenv()` 関数を使用して、環境変数の `AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY`、および `AWS_SESSION_TOKEN` を探しま

す。これらの認証情報は、環境の認証情報と呼ばれます。これらの値を取得する方法については、AWSSDK およびツールリファレンスガイドの「短期認証情報による認証」を参照してください。

AWS Elastic Beanstalk でアプリケーションをホストしている場合は、AWS Elastic Beanstalk コンソールを通じて AWS_ACCESS_KEY_ID、AWS_SECRET_KEY、および AWS_SESSION_TOKEN 環境変数を設定することで、これらの認証情報を SDK で自動的に使用できます。

環境変数の設定の詳細については、AWS SDK およびツールリファレンスガイドの「環境変数のサポート」を参照してください。また、ほとんどの AWS SDK でサポートされているすべての環境変数のリストについては、「環境変数リスト」を参照してください。

これらの環境変数は、次に示すように、コマンドラインで設定することもできます。

Linux

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
   # The access key for your AWS #####.
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   # The secret access key for your AWS #####.
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
   # The temporary session key for your AWS #####.
   # The AWS_SECURITY_TOKEN environment variable can also be used, but is only
 supported for backward compatibility purposes.
   # AWS_SESSION_TOKEN is supported by multiple AWS SDKs other than PHP.
```

Windows

```
C:\> SET  AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
   # The access key for your AWS #####.
C:\> SET  AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   # The secret access key for your AWS #####.
C:\> SET AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
   # The temporary session key for your AWS #####.
   # The AWS_SECURITY_TOKEN environment variable can also be used, but is only
 supported for backward compatibility purposes.
   # AWS_SESSION_TOKEN is supported by multiple AWS SDKs besides PHP.
```

IAM ロールの継承

IAM ロールによる Amazon EC2 インスタンス変数の認証情報の取得

Amazon EC2 インスタンスでアプリケーションを実行している場合、 を呼び出すための認証情報を提供するには、IAM ロール AWS を使用して一時的なセキュリティ認証情報を取得することをお勧めします。

IAM ロールを使用すると、アプリケーションで認証情報の管理について気にする必要がなくなります。Amazon EC2 インスタンスのメタデータサーバーから一時認証情報を取得することで、インスタンスはロールを「継承」できます。

一時認証情報 (インスタンスプロファイルの認証情報とも呼ばれます) は、ロールのポリシーで許可されているアクションとリソースへのアクセスを可能にします。IAM サービスに対してインスタンスを安全に認証してロールを継承すること、および取得したロールの認証情報を定期的に更新することの細かい段取りはすべて Amazon EC2 によって処理されます。これにより、ユーザーはほとんど何もしなくてもアプリケーションの安全性が保たれます。一時的なセキュリティ認証情報を受け入れるサービスのリストについては、IAM ユーザーガイドの「IAM と連携するAWS サービス」を参照してください。

> **Note**
>
> メタデータサービスから毎回取得することを回避するために、`Aws\CacheInterface` のインスタンスをクライアントコンストラクタの `'credentials'` オプションとして渡すことができます。これにより、SDK はキャッシュされているインスタンスプロファイルの認証情報を代わりに使用します。詳細については、「AWS SDK for PHP バージョン 3 の設定」を参照してください。

SDK を使用して Amazon EC2 アプリケーションを開発する方法の詳細については、AWS SDK およびツールリファレンスガイドの「Amazon EC2 インスタンス用の IAM ロールの使用」を参照してください。

IAM ロールの作成と Amazon EC2 インスタンスへの割り当て

1. IAM クライアントを作成します。

   インポート

```
require 'vendor/autoload.php';
```

```
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);
```

2. 使用するアクションとリソースへのアクセス許可を持つ IAM ロールを作成します。

   サンプルコード

```
$result = $client->createRole([
    'AssumeRolePolicyDocument' => 'IAM JSON Policy', // REQUIRED
    'Description' => 'Description of Role',
    'RoleName' => 'RoleName', // REQUIRED
]);
```

3. IAM インスタンスプロファイルを作成し、結果の Amazon リソースネーム (ARN) を保存します。

   > **Note**
   >
   > の代わりに IAM コンソールを使用する場合 AWS SDK for PHP、コンソールはインスタンスプロファイルを自動的に作成し、対応するロールと同じ名前を付けます。
   > サンプルコード

```
$IPN = 'InstanceProfileName';

$result = $client->createInstanceProfile([
    'InstanceProfileName' => $IPN ,
]);

$ARN = $result['Arn'];
$InstanceID =  $result['InstanceProfileId'];
```

4. Amazon EC2 クライアントを作成します。

   インポート

```
require 'vendor/autoload.php';

use Aws\Ec2\Ec2Client;
```

サンプルコード

```
$ec2Client = new Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
]);
```

5. 実行中または停止中の Amazon EC2 インスタンスにインスタンスプロファイルを追加します。IAM ロールのインスタンスプロファイル名を使用します。

   サンプルコード

```
 $result = $ec2Client->associateIamInstanceProfile([
    'IamInstanceProfile' => [
        'Arn' => $ARN,
        'Name' => $IPN,
    ],
    'InstanceId' => $InstanceID
]);
```

詳細については、Amazon EC2 ユーザーガイドの「Amazon EC2 の IAM ロール」を参照してください。

Amazon ECS タスク用の IAM ロールの使用

Amazon Elastic Container Service (Amazon ECS) のタスクは、IAM ロールを引き受けて AWS API コールを行うことができます。これにより、Amazon EC2 インスタンスプロファイルから Amazon EC2 インスタンスに認証情報を提供する場合と同じような方法で、アプリケーションで使用する認証情報を管理できます。

長期 AWS 認証情報を作成してコンテナに配布したり、Amazon EC2 インスタンスのロールを使用する代わりに、一時的な認証情報を使用する IAM ロールを ECS タスク定義または `RunTask` API オペレーションに関連付けることができます。

コンテナタスクが引き受けることができる IAM ロールの使用方法の詳細については、Amazon ECS 開発者ガイドの「タスク IAM ロール」トピックを参照してください。タスク定義で `taskRoleArn` 形式のタスク IAM ロールを使用する例については、Amazon ECS 開発者ガイドの「タスク定義の例」も参照してください。

別の で IAM ロールを引き受ける AWS アカウント

(AWS アカウント アカウント A) で作業し、別のアカウント (アカウント B) でロールを引き受けるときは、まずアカウント B で IAM ロールを作成する必要があります。このロールにより、アカウント (アカウント A) のエンティティがアカウント B で特定のアクションを実行できるようになります。クロスアカウントアクセスの詳細については、「チュートリアル: IAM ロールを使用して AWS アカウント間でアクセスを委任する」を参照してください。

アカウント B にロールを作成した後、ロールの ARN を記録します。この ARN は、アカウント A からロールを引き受けるときに使用します。ロールは、アカウント A のエンティティに関連付けられた AWS 認証情報を使用して引き受けます。

の認証情報を使用して AWS STS クライアントを作成します AWS アカウント。以下では認証情報プロファイルを使用しますが、任意の方法を使用できます。新しく作成した AWS STS クライアントで assume-role を呼び出し、カスタム sessionName を提供します。結果から新しい一時認証情報を取得します。認証情報の有効期間は 1 時間です。

サンプルコード

```
$stsClient = new Aws\Sts\StsClient([
    'profile' => 'default',
    'region' => 'us-east-2',
    'version' => '2011-06-15'
]);

$ARN = "arn:aws:iam::123456789012:role/xaccounts3access";
$sessionName = "s3-access-example";

$result = $stsClient->AssumeRole([
      'RoleArn' => $ARN,
      'RoleSessionName' => $sessionName,
]);

 $s3Client = new S3Client([
    'version'     => '2006-03-01',
    'region'      => 'us-west-2',
    'credentials' =>  [
```

```
        'key'    => $result['Credentials']['AccessKeyId'],
        'secret' => $result['Credentials']['SecretAccessKey'],
        'token'  => $result['Credentials']['SessionToken']
    ]
]);
```

詳細については、 AWS SDK for PHP API リファレンスの「IAM ロールの使用」または「AssumeRole」を参照してください。

ウェブ ID がある IAM ロールの使用

Web Identity フェデレーションを使用すると、お客様は AWS リソースにアクセスするときに認証にサードパーティーの ID プロバイダーを使用できます。ウェブ ID があるロールを継承できるようになるには、その前に IAM ロールを作成してウェブ ID プロバイダー (dP) を設定する必要があります。詳細については、「ウェブ ID または OpenID Connect フェデレーション用のロールを作成する (コンソール)」を参照してください。

ID プロバイダーを作成し、ウェブ ID のロールを作成したら、 クライアントを使用してユーザーを認証します。 AWS STS ID ProviderId の webIdentityToken と 、およびユーザーのアクセス許可を持つ IAM ロールのロール ARN を指定します。

サンプルコード

```
$stsClient = new Aws\Sts\StsClient([
    'profile' => 'default',
    'region' => 'us-east-2',
    'version' => '2011-06-15'
]);

$ARN = "arn:aws:iam::123456789012:role/xaccounts3access";
$sessionName = "s3-access-example";
$duration = 3600;

$result = $stsClient->AssumeRoleWithWebIdentity([
      'WebIdentityToken' => "FACEBOOK_ACCESS_TOKEN",
      'ProviderId' => "graph.facebook.com",
      'RoleArn' => $ARN,
      'RoleSessionName' => $sessionName,
]);

 $s3Client = new S3Client([
    'version'     => '2006-03-01',
```

```
    'region'      => 'us-west-2',
    'credentials' =>  [
        'key'    => $result['Credentials']['AccessKeyId'],
        'secret' => $result['Credentials']['SecretAccessKey'],
        'token'  => $result['Credentials']['SessionToken']
    ]
]);
```

詳細については、AssumeRoleWithWebIdentity「 ウェブベースの ID プロバイダー経由のフェデレーション」または AWS SDK for PHP 「 API リファレンスAssumeRoleWithWebIdentity」を参照してください。

プロファイルのあるロールを継承する

~/.aws/credentials でプロファイルを定義する

でプロファイルを定義することで、IAM ロール AWS SDK for PHP を使用するように を設定できます~/.aws/credentials。

継承するロールの role_arn 設定を持つ新しいプロファイルを作成します。また、IAM ロールを継承するアクセス許可を持つ認証情報のあるプロファイルの source_profile 設定を含めます。これらの設定について詳しくは、AWS SDK およびツールリファレンスガイドの「ロールの継承の認証情報」を参照してください。

たとえば、以下の ~/.aws/credentials では、project1 プロファイルが role_arn を設定し、default プロファイルを認証情報のソースとして指定して、そのプロファイルに関連付けられているエンティティがロールを引き継ぐことができることを確認しています。

```
[project1]
role_arn = arn:aws:iam::123456789012:role/testing
source_profile = default
role_session_name = OPTIONAL_SESSION_NAME

[default]
aws_access_key_id = YOUR_AWS_ACCESS_KEY_ID
aws_secret_access_key = YOUR_AWS_SECRET_ACCESS_KEY
aws_session_token= YOUR_AWS_SESSION_TOKEN
```

AWS_PROFILE 環境変数を設定するか、サービスクライアントをインスタンス化する際に profile パラメーターを使用することで、project1 で特定されたロールは、default プロファイルをソース認証情報を使用して継承されます。

次のスニペットは、S3Client コンストラクタでの profile パラメーターの使用方法を示しています。S3Client には、project1 プロファイルに関連付けられたロールに関連する権限が付与されます。

```
$s3 = new Aws\S3\S3Client([
    'region' => 'us-east-1',
    'version' => '2006-03-01',
    'profile' => 'project1'
]);
```

~/.aws/config でプロファイルを定義する

~/.aws/config ファイルには、継承するプロファイルを含めることもできます。環境変数 AWS_SDK_LOAD_NONDEFAULT_CONFIG を設定すると、SDK for PHP は config ファイルからプロファイルをロードします。AWS_SDK_LOAD_NONDEFAULT_CONFIG を設定すると、SDK は ~/.aws/config と ~/.aws/credentials の両方からプロファイルをロードします。~/.aws/credentials のプロファイルは最後にロードされ、~/.aws/config から同じ名前でロードされたプロファイルよりも優先されます。どちらの場所からロードされたプロファイルも、source_profile または引き受けるプロファイルとして使用できます。

次の例では、config ファイルに定義されている project1 プロファイルと credentials ファイル内の default プロファイルを使用しています。AWS_SDK_LOAD_NONDEFAULT_CONFIG も設定されています。

```
# Profile in ~/.aws/config.

[profile project1]
role_arn = arn:aws:iam::123456789012:role/testing
source_profile = default
role_session_name = OPTIONAL_SESSION_NAME
```

```
# Profile in ~/.aws/credentials.

[default]
aws_access_key_id = YOUR_AWS_ACCESS_KEY_ID
aws_secret_access_key = YOUR_AWS_SECRET_ACCESS_KEY
aws_session_token= YOUR_AWS_SESSION_TOKEN
```

次のスニペットが表示されている S3Client コンストラクタを実行すると、project1 プロファイルで定義されたロールは、default プロファイルに関連付けられた認証情報を使用して引き継がれます。

```
$s3 = new Aws\S3\S3Client([
    'region' => 'us-east-1',
    'version' => '2006-03-01',
    'profile' => 'project1'
]);
```

認証情報プロバイダを使用する

認証情報プロバイダは、GuzzleHttp\Promise\PromiseInterface インスタンスで満たすかまたは Aws\Credentials\CredentialsInterface で拒否した Aws\Exception\CredentialsException を返す関数です。認証情報プロバイダを使用して、認証情報を作成するための独自のカスタムロジックを実装したり、認証情報のロードを最適化したりできます。

認証情報プロバイダは、クライアントコンストラクタの credentials オプションで渡されます。認証情報プロバイダは非同期であり、API オペレーションが呼び出されるたびに遅延して評価されるようになっています。そのため、認証情報プロバイダ関数を SDK のクライアントコンストラクタに渡しても、その認証情報はすぐには検証されません。認証情報プロバイダから認証情報オブジェクトが返されない場合、API オペレーションは Aws\Exception\CredentialsException で拒否されます。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

// Use the default credential provider
$provider = CredentialProvider::defaultProvider();

// Pass the provider to the client
$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

SDK の組み込みプロバイダ

SDK には複数のプロバイダが組み込まれています。これらを任意のカスタムプロバイダと組み合わせて使用できます。 AWS SDKs「 SDK とツールリファレンスガイド」の「標準化された認証情報プロバイダー」を参照してください。 AWS SDKs

> **Important**
>
> 認証情報プロバイダは、API 操作が実行されるたびに呼び出されます。認証情報をロードすることが負荷の高いタスク (たとえば、ディスクまたはネットワークリソースからのロード) である場合や、認証情報がプロバイダでキャッシュされない場合は、認証情報プロバイダを `Aws\Credentials\CredentialProvider::memoize` 関数内にラップすることを検討します。SDK で使用されるデフォルトの認証情報プロバイダは自動的にメモ化されます。

assumeRole プロバイダ

`Aws\Credentials\AssumeRoleCredentialProvider` を使用してロールを継承することによって認証情報を作成する場合は、次に示すように、`'client'` オブジェクトと `StsClient` の詳細を使用して `'assume_role_params'` 情報を提供する必要があります。

> **Note**
>
> API オペレーション AWS STS ごとに認証情報を不必要に取得しないようにするには、`memoize`関数を使用して、有効期限が切れたときに認証情報を自動的に更新する処理を行います。次のコードを例として参照してください。

```
use Aws\Credentials\CredentialProvider;
use Aws\Credentials\InstanceProfileProvider;
use Aws\Credentials\AssumeRoleCredentialProvider;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

// Passing Aws\Credentials\AssumeRoleCredentialProvider options directly
$profile = new InstanceProfileProvider();
$ARN = "arn:aws:iam::123456789012:role/xaccounts3access";
$sessionName = "s3-access-example";

$assumeRoleCredentials = new AssumeRoleCredentialProvider([
```

```
    'client' => new StsClient([
        'region' => 'us-east-2',
        'version' => '2011-06-15',
        'credentials' => $profile
    ]),
    'assume_role_params' => [
        'RoleArn' => $ARN,
        'RoleSessionName' => $sessionName,
    ],
]);

// To avoid unnecessarily fetching STS credentials on every API operation,
// the memoize function handles automatically refreshing the credentials when they
 expire
$provider = CredentialProvider::memoize($assumeRoleCredentials);

$client = new S3Client([
    'region'      => 'us-east-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

の詳細については'assume_role_params'、「」を参照してくださいAssumeRole。

SSO プロバイダー

Aws\Credentials\CredentialProvider::sso はシングルサインオン認証情報プロバイダーです。このプロバイダーは、 AWS IAM Identity Center 認証情報プロバイダーとも呼ばれます。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$credentials = new Aws\CredentialProvider::sso('profile default');

$s3 = new Aws\S3\S3Client([
    'version'     => 'latest',
    'region'      => 'us-west-2',
    'credentials' => $credentials
]);
```

名前付きプロファイルを使用する場合は、前の例の「default」をプロファイル名に置き換えてください。名前付きプロファイルの設定について詳しくは、AWS SDK およびツールリファレンスガイ

ドの「共有 config と credentials ファイル」を参照してください。または、AWS_PROFILE 環境変数を使用して、使用するプロファイルの設定を指定することもできます。

IAM ID センタープロバイダの仕組みについて詳しくは、AWS SDK およびツールリファレンスガイドの「IAM ID センターの認証について」を参照してください。

プロバイダチェーンの作成

Aws\Credentials\CredentialProvider::chain( ) 関数を使用して、認証情報プロバイダーチェーンを作成できます。この関数は、それぞれが認証情報プロバイダ関数である可変個引数を受け入れます。次に、この関数は、複数の指定した関数を合成した 1 つの新しい関数を返します (各関数は、いずれかのプロバイダから正常に満たされた promise が返されるまで順に呼び出されます)。

defaultProvider は、その合成関数を使用して、失敗するまで複数のプロバイダをチェックします。以下の defaultProvider のソースは chain 関数の使用例を示しています。

```
// This function returns a provider
public static function defaultProvider(array $config = [])
{
    // This function is the provider, which is actually the composition
    // of multiple providers. Notice that we are also memoizing the result by
    // default.
    return self::memoize(
        self::chain(
            self::env(),
            self::ini(),
            self::instanceProfile($config)
        )
    );
}
```

カスタムプロバイダの作成

認証情報プロバイダは、呼び出されたときに、GuzzleHttp\Promise\PromiseInterface で満たすかまたはAws\Credentials\CredentialsInterface で拒否した promise (Aws\Exception\CredentialsException) を返す単純な関数です。

プロバイダを作成する際のベストプラクティスとして、実際の認証情報プロバイダを作成するために呼び出される関数を作成します。その例として、env プロバイダのソース (例として使用するために多少変更している) を次に示します。この関数は、実際のプロバイダ関数を返す関数であることがわ

かります。そうすることによって、認証情報プロバイダの作成およびそれを値として渡すことが簡単になります。

```
use GuzzleHttp\Promise;
use GuzzleHttp\Promise\RejectedPromise;

// This function CREATES a credential provider
public static function env()
{
    // This function IS the credential provider
    return function () {
        // Use credentials from environment variables, if available
        $key = getenv(self::ENV_KEY);
        $secret = getenv(self::ENV_SECRET);
        if ($key && $secret) {
            return Promise\promise_for(
                new Credentials($key, $secret, getenv(self::ENV_SESSION))
            );
        }

        $msg = 'Could not find environment variable '
            . 'credentials in ' . self::ENV_KEY . '/' . self::ENV_SECRET;
        return new RejectedPromise(new CredentialsException($msg));
    };
}
```

defaultProvider プロバイダ

`Aws\Credentials\CredentialProvider::defaultProvider` はデフォルトの認証情報プロバイダです。このプロバイダは、クライアントの作成時に `credentials` オプションを指定しなかった場合に使用されます。このプロバイダは、環境変数、.ini ファイル (`.aws/credentials` ファイルから `.aws/config` ファイルの順)、インスタンスプロファイル (`EcsCredentials` から `Ec2` メタデータの順) の順で認証情報のロードを試行します。

> **Note**
>
> デフォルトのプロバイダの結果は自動的にメモ化されます。

ecsCredentials プロバイダ

`Aws\Credentials\CredentialProvider::ecsCredentials` は、GET リクエストによる認証情報のロードを試行します。その URI はコンテナの `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` 環境変数で指定します。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$provider = CredentialProvider::ecsCredentials();
// Be sure to memoize the credentials
$memoizedProvider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $memoizedProvider
]);
```

env プロバイダ

`Aws\Credentials\CredentialProvider::env` は、環境変数からの認証情報のロードを試行します。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => CredentialProvider::env()
]);
```

assumeRoleWithWebIdentityCredentialProvider プロバイダー

`Aws\Credentials\CredentialProvider::assumeRoleWithWebIdentityCredentialProvider` は、ロールを引き受けることで認証情報のロードを試みます。環境変数 `AWS_ROLE_ARN` および `AWS_WEB_IDENTITY_TOKEN_FILE` が存在する場合、プロバイダーは、`AWS_WEB_IDENTITY_TOKEN_FILE` で指定されたフルパスのディスク上のトークンを使用し

て、AWS_ROLE_ARN で指定されたロールの引き受けを試みます。環境変数が使用された場合、プロバイダーは AWS_ROLE_SESSION_NAME 環境変数からセッションの設定を試みます。

環境変数が設定されていない場合、プロバイダーはデフォルトのプロファイル、または AWS_PROFILE として設定されているプロファイルを使用します。デフォルトでは、プロバイダーは ~/.aws/credentials および ~/.aws/config からプロファイルを読み取り、filename config オプションで指定されたプロファイルから読み取ることができます。プロバイダーはプロファイルの role_arn ロールを引き受け、web_identity_token_file で設定されたフルパスからトークンを読み取ります。プロファイルで設定されている場合は、role_session_name が使用されます。

デフォルトのチェーンの一部としてプロバイダーが呼び出され、直接呼び出すことができます。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$provider = CredentialProvider::assumeRoleWithWebIdentityCredentialProvider();
// Cache the results in a memoize function to avoid loading and parsing
// the ini file on every API operation
$provider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

デフォルトでは、この認証情報プロバイダーは、 がロールを引き受け StsClient るために使用する設定済みリージョンを継承します。オプションで、完全な StsClient を指定できます。認証情報は、提供された falseで として設定する必要があります StsClient。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$stsClient = new StsClient([
    'region'      => 'us-west-2',
    'version'     => 'latest',
    'credentials' => false
])

$provider = CredentialProvider::assumeRoleWithWebIdentityCredentialProvider([
```

```
    'stsClient' => $stsClient
]);
// Cache the results in a memoize function to avoid loading and parsing
// the ini file on every API operation
$provider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

ini プロバイダ

`Aws\Credentials\CredentialProvider::ini` は、.ini 認証情報ファイルからの認証情報もロードを試行します。デフォルトでは、SDK は にある共有 AWS `credentials`ファイルから「デフォルト」プロファイルをロードしようとします`~/.aws/credentials`。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$provider = CredentialProvider::ini();
// Cache the results in a memoize function to avoid loading and parsing
// the ini file on every API operation
$provider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

プロバイダを作成する関数に引数を指定することによって、カスタムプロファイルまたは .ini ファイルの場所を指定できます。

```
$profile = 'production';
$path = '/full/path/to/credentials.ini';

$provider = CredentialProvider::ini($profile, $path);
$provider = CredentialProvider::memoize($provider);

$client = new S3Client([
```

```
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

プロセスプロバイダ

`Aws\Credentials\CredentialProvider::process` は、ini 認証情報ファイルで指定された credential_process から、認証情報のロードを試みます。デフォルトでは、SDK は にある共有 AWS `credentials`ファイルから「デフォルト」プロファイルをロードしようとします`~/.aws/credentials`。SDK は指定どおりに credential_process コマンドを呼び出し、stdout から JSON データを読み取ります。credential_process は次の形式で stdout に認証情報を書き込む必要があります。

```
{
    "Version": 1,
    "AccessKeyId": "",
    "SecretAccessKey": "",
    "SessionToken": "",
    "Expiration": ""
}
```

`SessionToken` および `Expiration` はオプションです。存在する場合、認証情報は一時的なものとして扱われます。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$provider = CredentialProvider::process();
// Cache the results in a memoize function to avoid loading and parsing
// the ini file on every API operation
$provider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

プロバイダを作成する関数に引数を指定することによって、カスタムプロファイルまたは .ini ファイルの場所を指定できます。

```
$profile = 'production';
$path = '/full/path/to/credentials.ini';

$provider = CredentialProvider::process($profile, $path);
$provider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

instanceProfile プロバイダ

`Aws\Credentials\CredentialProvider::instanceProfile` は、Amazon EC2 インスタンスプロファイルからの認証情報のロードを試行します。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$provider = CredentialProvider::instanceProfile();
// Be sure to memoize the credentials
$memoizedProvider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $memoizedProvider
]);
```

デフォルトでは、プロバイダーは最大 3 回まで認証情報の取得を試みます。再試行の数は `retries` オプションで設定でき、このオプションを `0` に設定することで完全に無効にできます。

```
use Aws\Credentials\CredentialProvider;

$provider = CredentialProvider::instanceProfile([
    'retries' => 0
]);
$memoizedProvider = CredentialProvider::memoize($provider);
```

> **Note**
>
> AWS_EC2_METADATA_DISABLED 環境変数を true に設定することによって、Amazon EC2 インスタンスプロファイルからのロードの試行を無効にできます。

認証情報のメモ化

前回の戻り値を記憶している認証情報プロバイダの作成が必要になる場合があります。そのような認証情報プロバイダは、認証情報のロードが負荷の高いオペレーションである場合や、Aws\Sdk クラスを使用して複数のクライアントで 1 つの認証情報プロバイダを共有にする場合に、パフォーマンスを向上させるために便利であることがあります。認証情報プロバイダ関数を memoize 関数内にラップすることによって、認証情報プロバイダにメモ化を追加できます。

```
use Aws\Credentials\CredentialProvider;

$provider = CredentialProvider::instanceProfile();
// Wrap the actual provider in a memoize function
$provider = CredentialProvider::memoize($provider);

// Pass the provider into the Sdk class and share the provider
// across multiple clients. Each time a new client is constructed,
// it will use the previously returned credentials as long as
// they haven't yet expired.
$sdk = new Aws\Sdk(['credentials' => $provider]);

$s3 = $sdk->getS3(['region' => 'us-west-2', 'version' => 'latest']);
$ec2 = $sdk->getEc2(['region' => 'us-west-2', 'version' => 'latest']);

assert($s3->getCredentials() === $ec2->getCredentials());
```

メモ化された認証情報の有効期限が切れると、memoize ラッパーはラップされたプロバイダを呼び出して認証情報の更新を試行します。

から一時的な認証情報を使用する AWS STS

AWS Security Token Service （ AWS STS) では、IAM ユーザー、または ID フェデレーションを介して認証するユーザーの制限付き権限、一時的な認証情報をリクエストできます。理解を深めるには、IAM ユーザーガイドの「IAM の一時的なセキュリティ認証情報」を参照してください。一時的なセキュリティ認証情報を使用して、ほとんどの AWS サービスにアクセスできます。一時的なセ

キュリティ認証情報を受け入れるサービスのリストについては、IAM ユーザーガイドの「IAM と連携するAWS サービス」を参照してください。

一時的な認証情報の一般的なユースケースの 1 つは、サードパーティーの ID プロバイダーを介してユーザーを認証することで、モバイルまたはクライアント側のアプリケーションに AWS リソースへのアクセスを許可することです (「ウェブアイデンティティフェデレーション」を参照) 。

一時的な認証情報を取得する

AWS STS には一時的な認証情報を返すオペレーションがいくつかありますが、そのGetSessionTokenオペレーションは最も簡単にデモンストレーションできます。次のスニペットは、PHP SDK の STS クライアントの getSessionTokenメソッドを呼び出して一時的な認証情報を取得します。

```
$sdk = new Aws\Sdk([
    'region'   => 'us-east-1',
]);

$stsClient = $sdk->createSts();

$result = $stsClient->getSessionToken();
```

GetSessionToken およびその他のオペレーションの結果に AWS STS は常に'Credentials'値が含まれます。を $result (例えば を使用してprint_r($result)) 印刷すると、次のようになります。

```
Array
(
    ...
    [Credentials] => Array
    (
        [SessionToken] => '<base64 encoded session token value>'
        [SecretAccessKey] => '<temporary secret access key value>'
        [Expiration] => 2013-11-01T01:57:52Z
        [AccessKeyId] => '<temporary access key value>'
    )
    ...
)
```

への一時的な認証情報の提供 AWS SDK for PHP

AWS クライアントをインスタンス化し、 から直接受け取った値を渡すことで、別のクライアントで一時的な認証情報を使用できます AWS STS 。

```
use Aws\S3\S3Client;

$result = $stsClient->getSessionToken();

$s3Client = new S3Client([
    'version'     => '2006-03-01',
    'region'      => 'us-west-2',
    'credentials' => [
        'key'    => $result['Credentials']['AccessKeyId'],
        'secret' => $result['Credentials']['SecretAccessKey'],
        'token'  => $result['Credentials']['SessionToken']
    ]
]);
```

Aws\Credentials\Credentials オブジェクトを作成し、これをクライアントをインスタンス化するときに使用することもできます。

```
use Aws\Credentials\Credentials;
use Aws\S3\S3Client;

$result = $stsClient->getSessionToken();

$credentials = new Credentials(
    $result['Credentials']['AccessKeyId'],
    $result['Credentials']['SecretAccessKey'],
    $result['Credentials']['SessionToken']
);

$s3Client = new S3Client([
    'version'     => '2006-03-01',
    'region'      => 'us-west-2',
    'credentials' => $credentials
]);
```

ただし、一時認証情報を提供するための最良の方法は、StsClient に含まれている createCredentials() ヘルパーメソッドを使用することです。このメソッドは、 AWS STS 結果からデータを抽出し、Credentialsオブジェクトを作成します。

```
$result = $stsClient->getSessionToken();
$credentials = $stsClient->createCredentials($result);

$s3Client = new S3Client([
    'version'     => '2006-03-01',
    'region'      => 'us-west-2',
    'credentials' => $credentials
]);
```

アプリケーションまたはプロジェクトで一時的な認証情報を使用する必要がある理由の詳細については、 AWS STS ドキュメントの「一時的なアクセスを許可するシナリオ」を参照してください。

匿名クライアントを作成する

場合によっては、認証情報が関連付けられていないクライアントを作成する必要があります。そのようなクライアントを使用すると、サービスへの匿名リクエストを行うことができます。

例えば、匿名アクセスを許可するように Amazon S3 オブジェクトと Amazon CloudSearch ドメインの両方を設定できます。

匿名クライアントを作成するには、`'credentials'` オプションを `false` に設定します。

```
$s3Client = new S3Client([
    'version'     => 'latest',
    'region'      => 'us-west-2',
    'credentials' => false
]);

// Makes an anonymous request. The object would need to be publicly
// readable for this to succeed.
$result = $s3Client->getObject([
    'Bucket' => 'my-bucket',
    'Key'    => 'my-key',
]);
```

AWS SDK for PHP バージョン 3 でのコマンドオブジェクト

AWS SDK for PHP はコマンドパターンを使用して、後から HTTP リクエストを転送するために使用するパラメーターとハンドラーをカプセル化します。

コマンドの暗黙的な使用

任意のクライアントクラスをテストすると、API オペレーションに対応するメソッドが実際には存在しないことを確認できます。これらは `__call()` マジックメソッドを使用して実装されます。これらの疑似メソッドは、実際は SDK でのコマンドオブジェクトの使用をカプセル化するショートカットです。

通常、コマンドオブジェクトを直接操作する必要はありません。`Aws\S3\S3Client::putObject()` のようなメソッドを呼び出すと、SDK は実際には指定されたパラメーターに基づいて `Aws\CommandInterface` オブジェクトを作成し、コマンドを実行して、データが入力されている `Aws\ResultInterface` オブジェクトを返します (またはエラーで例外がスローされます)。クライアントのいずれかの `Async` メソッド (例 :`Aws\S3\S3Client::putObjectAsync()`) を呼び出すときにも、同様の流れになります。クライアントは、指定されたパラメーターに基づいてコマンドを作成し、HTTP リクエストをシリアル化して、リクエストを開始し、promise を返します。

以下の例も機能的に同様です。

```
$s3Client = new Aws\S3\S3Client([
    'version' => '2006-03-01',
    'region'  => 'us-standard'
]);

$params = [
    'Bucket' => 'foo',
    'Key'    => 'baz',
    'Body'   => 'bar'
];

// Using operation methods creates a command implicitly
$result = $s3Client->putObject($params);

// Using commands explicitly
$command = $s3Client->getCommand('PutObject', $params);
$result = $s3Client->execute($command);
```

コマンドのパラメーター

すべてのコマンドは、サービスの API の一部ではなく、SDK の動作を制御する特別なパラメータをいくつかサポートしてします。

@http

このパラメータを使用すると、基になる HTTP ハンドラーがリクエストを実行する方法を微調整できます。@http パラメータに含めることができるオプションは、「http」クライアントオプションを使用してそのクライアントをインスタンス化するときに設定できるオプションと同じです。

```
// Configures the command to be delayed by 500 milliseconds
$command['@http'] = [
    'delay' => 500,
];
```

@retries

「retries」クライアントオプションと同様に、@retries は、コマンドが失敗したと見なされるまでに再試行できる回数を制御します。再試行を無効にするには、0 に設定します。

```
// Disable retries
$command['@retries'] = 0;
```

> **Note**
>
> クライアントで再試行を無効にしている場合は、そのクライアントに渡される個別のコマンドで再試行を選択的に有効にすることはできません。

コマンドオブジェクトの作成

クライアントの getCommand() メソッドを使用してコマンドを作成することができます。HTTP リクエストは、すぐに実行または転送されず、クライアントの execute() メソッドに渡されたときにのみ実行されます。これにより、コマンドを実行する前にコマンドオブジェクトを変更する機会が得られます。

```
$command = $s3Client->getCommand('ListObjects');
$command['MaxKeys'] = 50;
$command['Prefix'] = 'foo/baz/';
$result = $s3Client->execute($command);

// You can also modify parameters
$command = $s3Client->getCommand('ListObjects', [
    'MaxKeys' => 50,
```

```
    'Prefix'  => 'foo/baz/',
]);
$command['MaxKeys'] = 100;
$result = $s3Client->execute($command);
```

コマンド

コマンドがクライアントから作成された場合、クライアントの Aws\HandlerList オブジェクトのクローンが与えられます。コマンドは、クライアントのハンドラーリストのクローンとして与えられ、クライアントが実行する他のコマンドに影響しない、カスタムミドルウェアとハンドラーを使用するためにコマンドが許可されます。

つまり、(例 :Aws\MockHandler) コマンドごとに別の HTTP クライアントを使用し、ミドルウェアからコマンドごとのカスタム動作を追加できます。次の例では、MockHandler を使用して、実際の HTTP リクエストを送信する代わりに疑似の結果を作成します。

```
use Aws\Result;
use Aws\MockHandler;

// Create a mock handler
$mock = new MockHandler();
// Enqueue a mock result to the handler
$mock->append(new Result(['foo' => 'bar']));
// Create a "ListObjects" command
$command = $s3Client->getCommand('ListObjects');
// Associate the mock handler with the command
$command->getHandlerList()->setHandler($mock);
// Executing the command will use the mock handler, which returns the
// mocked result object
$result = $client->execute($command);

echo $result['foo']; // Outputs 'bar'
```

コマンドを使用するハンドラーの変更に加えて、カスタムミドルウェアをコマンドに挿入することもできます。次の例では、ハンドラーリストでオブザーバーとして機能する tap ミドルウェアを使用します。

```
use Aws\CommandInterface;
use Aws\Middleware;
use Psr\Http\Message\RequestInterface;
```

```
$command = $s3Client->getCommand('ListObjects');
$list = $command->getHandlerList();

// Create a middleware that just dumps the command and request that is
// about to be sent
$middleware = Middleware::tap(
    function (CommandInterface $command, RequestInterface $request) {
        var_dump($command->toArray());
        var_dump($request);
    }
);

// Append the middleware to the "sign" step of the handler list. The sign
// step is the last step before transferring an HTTP request.
$list->append('sign', $middleware);

// Now transfer the command and see the var_dump data
$s3Client->execute($command);
```

CommandPool

Aws\CommandPool オブジェクトを生成するイテレーターを使用して、Aws\CommandInterface により、コマンドを同時に実行できます。CommandPool は、プール内のコマンドを繰り返し実行する (コマンドが完了時に、固定プールサイズを確保するために複数実行される) 間に、一定数のコマンドが同時に実行されるようにします。

ここでは、CommandPool を使用し、いくつかのコマンドを送信するだけの非常に簡単な例を示します。

```
use Aws\S3\S3Client;
use Aws\CommandPool;

// Create the client
$client = new S3Client([
    'region'  => 'us-standard',
    'version' => '2006-03-01'
]);

$bucket = 'example';
$commands = [
    $client->getCommand('HeadObject', ['Bucket' => $bucket, 'Key' => 'a']),
    $client->getCommand('HeadObject', ['Bucket' => $bucket, 'Key' => 'b']),
```

```
    $client->getCommand('HeadObject', ['Bucket' => $bucket, 'Key' => 'c'])
];

$pool = new CommandPool($client, $commands);

// Initiate the pool transfers
$promise = $pool->promise();

// Force the pool to complete synchronously
$promise->wait();
```

この例では CommandPool のごく一部の機能しか利用していません。より複雑な例を試してみましょう。例えば、ディスク上のファイルを Amazon S3 バケットにアップロードするとします。ディスクからファイルのリストを取得するには、PHP の DirectoryIterator を使用できます。このイテレーターにより SplFileInfo オブジェクトが生成されます。CommandPool は Aws\CommandInterface オブジェクトを生成するイテレーターを受け入れます。そのため、SplFileInfo オブジェクトにマッピングして Aws\CommandInterface オブジェクトを返します。

```
<?php
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\S3\S3Client;
use Aws\CommandPool;
use Aws\CommandInterface;
use Aws\ResultInterface;
use GuzzleHttp\Promise\PromiseInterface;

// Create the client
$client = new S3Client([
    'region'  => 'us-standard',
    'version' => '2006-03-01'
]);

$fromDir = '/path/to/dir';
$toBucket = 'my-bucket';

// Create an iterator that yields files from a directory
$files = new DirectoryIterator($fromDir);

// Create a generator that converts the SplFileInfo objects into
```

```
// Aws\CommandInterface objects. This generator accepts the iterator that
// yields files and the name of the bucket to upload the files to.
$commandGenerator = function (\Iterator $files, $bucket) use ($client) {
    foreach ($files as $file) {
        // Skip "." and ".." files
        if ($file->isDot()) {
            continue;
        }
        $filename = $file->getPath() . '/' . $file->getFilename();
        // Yield a command that is executed by the pool
        yield $client->getCommand('PutObject', [
            'Bucket' => $bucket,
            'Key'    => $file->getBaseName(),
            'Body'   => fopen($filename, 'r')
        ]);
    }
};

// Now create the generator using the files iterator
$commands = $commandGenerator($files, $toBucket);

// Create a pool and provide an optional array of configuration
$pool = new CommandPool($client, $commands, [
    // Only send 5 files at a time (this is set to 25 by default)
    'concurrency' => 5,
    // Invoke this function before executing each command
    'before' => function (CommandInterface $cmd, $iterKey) {
        echo "About to send {$iterKey}: "
            . print_r($cmd->toArray(), true) . "\n";
    },
    // Invoke this function for each successful transfer
    'fulfilled' => function (
        ResultInterface $result,
        $iterKey,
        PromiseInterface $aggregatePromise
    ) {
        echo "Completed {$iterKey}: {$result}\n";
    },
    // Invoke this function for each failed transfer
    'rejected' => function (
        AwsException $reason,
        $iterKey,
        PromiseInterface $aggregatePromise
    ) {
```

```
        echo "Failed {$iterKey}: {$reason}\n";
    },
]);

// Initiate the pool transfers
$promise = $pool->promise();

// Force the pool to complete synchronously
$promise->wait();

// Or you can chain the calls off of the pool
$promise->then(function() { echo "Done\n"; });
```

CommandPool の設定

Aws\CommandPool コンストラクタはさまざまな設定オプションを受け入れます。

concurrency (callable|int)

同時に実行するコマンドの最大数。プールを動的にサイズ変更する関数を提供します。この関数は、現在保留中のリクエスト数を取得し、新しいプールのサイズ制限を表す整数を返します。

before (callable)

各コマンドを送信する前に呼び出す関数。この before 関数では、コマンドとコマンドのイテレーターのキーを受け付けます。コマンドを送信する前に、before 関数で、必要に応じてコマンドを変更できます。

fulfilled (callable)

promise が実行されたときに呼び出す関数。この関数は、プールを省略する必要がある場合、解決または拒否できる集計 promise と結果の基になるイテレーターの ID、結果オブジェクトが提供されます。

rejected (callable)

promise が拒否されたときに呼び出す関数。この関数は、プールを省略する必要がある場合、解決または拒否できる集計 promise と例外の基になるイテレーターの ID、Aws\Exception オブジェクトが提供されます。

コマンド間の手動のガベージコレクション

大きなコマンドプールでメモリ制限に達している場合、メモリ制限に達したときに PHP ガベージコレクターでまだ収集されていない SDK によって生成された巡回参照が原因である可能性があります。コマンド間で収集アルゴリズムを手動で呼び出すと、制限に達する前にサイクルを収集できます。次の例では、各コマンドを送信する前にコールバックを使用して収集アルゴリズムを呼び出す CommandPool を作成します。ガベージコレクターを呼び出してもパフォーマンスに影響することはなく、最適な使用はお客様のユースケースと環境によって異なることに注意してください。

```
$pool = new CommandPool($client, $commands, [
    'concurrency' => 25,
    'before' => function (CommandInterface $cmd, $iterKey) {
        gc_collect_cycles();
    }
]);
```

AWS SDK for PHP バージョン 3 の promise

AWS SDK for PHP では promise を使用して非同期ワークフローの使用を許可します。この非同期性により、HTTP リクエストの同時送信が可能になります。SDK によって使用される promise の仕様は Promises/A+ です。

promise とは何ですか?

promise は、非同期オペレーションの最終結果を表します。promise を操作する主な方法は、その then メソッドを介するものです。このメソッドは、promise の最終結果または実行できない理由を受け取るコールバックを登録します。

AWS SDK for PHP は、promise の実装を guzzlehttp/promises Composer パッケージに依存します。Guzzle promise はブロッキングとノンブロッキングワークフローをサポートし、ノンブロッキングループで使用できます。

> **Note**
>
> AWS SDK for PHP で 1 つのスレッドを使用して、HTTP リクエストが同時に送信されます。ノンブロッキング呼び出しは、状態の変更 (promise の実行または拒否など) に対応する間に、1 つ以上の HTTP リクエストを転送するために使用されます。

SDK での promise

promise は、SDK 全体で使用されます。例えば、promise は SDK で提供されるほとんどの高レベル抽象化、ページネーター、ウェーター、コマンドプール、マルチパートアップロード、S3 ディレクトリ/バケット転送などで使用されます。

SDK が提供するすべてのクライアントは、Async が付いたメソッドのいずれかを呼び出した場合に promise を返します。例えば、次のコードでは、Amazon DynamoDB DescribeTable オペレーションの結果を取得するために promise を作成する方法を示しています。

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'region'  => 'us-west-2',
    'version' => 'latest',
]);

// This will create a promise that will eventually contain a result
$promise = $client->describeTableAsync(['TableName' => 'mytable']);
```

describeTable または describeTableAsync のどちらかを呼び出すことができることに注意してください。これらのメソッドは、クライアントのマジック __call メソッドで、クライアントに関連付けられている version 番号と API モデルで動いています。describeTable のような Async が付いていないメソッドを呼び出すことにより、HTTP リクエストを送信する間、クライアントはブロックし、Aws\ResultInterface オブジェクトを返すか Aws\Exception\AwsException をスローします。オペレーション名の後ろに Async を付けると (describeTableAsync など)、クライアントは promise を作成します。これは最終的に実行されると Aws\ResultInterface オブジェクト、または拒否されると Aws\Exception\AwsException になります。

> **Important**
>
> promise が返されるとき、結果がすでに到着しています (たとえば、モックハンドラーを使用する場合)、または HTTP リクエストが開始されません。

then メソッドを使用して、promise とコールバックを登録できます。このメソッドは、2 つのコールバック、$onFulfilled と $onRejected (どちらも省略可能) を受け付けます。promise が実行される場合に、$onFulfilled コールバックが呼び出され、promise が拒否される (失敗した) 場合に、$onRejected コールバックが呼び出されます。

```
$promise->then(
    function ($value) {
        echo "The promise was fulfilled with {$value}";
    },
    function ($reason) {
        echo "The promise was rejected with {$reason}";
    }
);
```

コマンドの同時実行

複数の promise は同時に実行されるようにまとめて構成できます。これを行うには、SDK とノンブロッキングイベントループを統合する、複数の promise を作成し、同時に完了するまで待機します。

```
use GuzzleHttp\Promise\Utils;

$sdk = new Aws\Sdk([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

$s3 = $sdk->createS3();
$ddb = $sdk->createDynamoDb();

$promises = [
    'buckets' => $s3->listBucketsAsync(),
    'tables'  => $ddb->listTablesAsync(),
];

// Wait for both promises to complete.
$results = Utils::unwrap($promises);

// Notice that this method will maintain the input array keys.
var_dump($results['buckets']->toArray());
var_dump($results['tables']->toArray());
```

> **Note**
>
> CommandPool には複数の API オペレーションを同時に実行するためのさらに強力なメカニズムが用意されています。

promise の連鎖

promise のすばらしい特徴の 1 つは、組み合わせられること、つまり変換パイプラインを作成することができます。複数の promise は then コールバックと後続の then コールバックを連結することで構成されます。then メソッドの戻り値が promise です。指定したコールバックの結果に基づいて、実行されるか拒否されます。

```
$promise = $client->describeTableAsync(['TableName' => 'mytable']);

$promise
    ->then(
        function ($value) {
            $value['AddedAttribute'] = 'foo';
            return $value;
        },
        function ($reason) use ($client) {
            // The call failed. You can recover from the error here and
            // return a value that will be provided to the next successful
            // then() callback. Let's retry the call.
            return $client->describeTableAsync(['TableName' => 'mytable']);
        }
    )->then(
        function ($value) {
            // This is only invoked when the previous then callback is
            // fulfilled. If the previous callback returned a promise, then
            // this callback is invoked only after that promise is
            // fulfilled.
            echo $value['AddedAttribute']; // outputs "foo"
        },
        function ($reason) {
            // The previous callback was rejected (failed).
        }
    );
```

> **Note**
>
> promise コールバックの戻り値は $value 引数で、ダウンストリームの promise に渡されます。ダウンストリームの promise チェーンに値を指定する場合は、コールバック関数に値を返す必要があります。

拒否の転送

promise が拒否されたときに呼び出すコールバックを登録することができます。任意のコールバックで例外がスローされた場合、promise は例外で拒否され、チェーンの次の promise も例外で拒否されます。`$onRejected` コールバックから正常な値を返す場合、promise チェーンの次の promise が `$onRejected` コールバックからの戻り値で実行されます。

promise の待機

promise の `wait` メソッドを使用すると、promise が同期的に完了するように強制できます。

```
$promise = $client->listTablesAsync();
$result = $promise->wait();
```

promise の `wait` 関数を呼び出している間に例外が発生した場合は、例外により promise が拒否され、例外がスローされます。

```
use Aws\Exception\AwsException;

$promise = $client->listTablesAsync();

try {
    $result = $promise->wait();
} catch (AwsException $e) {
    // Handle the error
}
```

実行された promise での `wait` の呼び出しは wait 関数をトリガーしません。以前に配信された値だけを返します。

```
$promise = $client->listTablesAsync();
$result = $promise->wait();
assert($result ### $promise->wait());
```

拒否された promise での `wait` の呼び出しでは例外がスローされます。拒否理由が `\Exception` のインスタンスである場合、その理由がスローされます。それ以外の場合は、`GuzzleHttp\Promise\RejectionException` がスローされ、例外の `getReason` メソッドを呼び出して理由を取得できます。

> **Note**
>
> AWS SDK for PHP での API オペレーションの呼び出しは `Aws\Exception\AwsException` クラスのサブクラスで拒否されます。ただし、then メソッドが配信する理由が異なる場合もあります。拒否理由を変更するカスタムミドルウェアを追加できるためです。

promise のキャンセル

promise の `cancel()` メソッドを使用して、promise をキャンセルできます。promise がすでに解決された場合、`cancel()` を呼び出しても何も起きません。promise のキャンセルは、promise 自体とそこからの配信を待機している promise をキャンセルします。キャンセルされた promise は `GuzzleHttp\Promise\RejectionException` で拒否されます。

promise の結合

promise を組み合わせて複合 promise を作成し、複雑なワークフローを構築できます。`guzzlehttp/promise` パッケージには、promise を結合するために使用できるさまざまな関数が含まれています。

promise コレクション関数のすべての API ドキュメントを見つけるには、namespace-GuzzleHttp.Promise を参照してください。

each および each_limit

`Aws\CommandInterface` コマンドのタスクキューがあるとき、CommandPool を使用して、固定プールサイズ (コマンドをメモリに格納または遅延イテレーターで出力可能) で同時に実行します。`CommandPool` により、固定数のコマンドは、指定されたイテレーターを使い果たすまで同時に送信されます。

`CommandPool` は、同じクライアントで実行されたコマンドの場合にのみ機能します。固定プールサイズを使用して、異なるクライアントにコマンドを同時に送信するために、`GuzzleHttp\Promise\each_limit` 関数を使用できます。

```
use GuzzleHttp\Promise;

$sdk = new Aws\Sdk([
    'version' => 'latest',
```

```
    'region'  => 'us-west-2'
]);

$s3 = $sdk->createS3();
$ddb = $sdk->createDynamoDb();

// Create a generator that yields promises
$promiseGenerator = function () use ($s3, $ddb) {
    yield $s3->listBucketsAsync();
    yield $ddb->listTablesAsync();
    // yield other promises as needed...
};

// Execute the tasks yielded by the generator concurrently while limiting the
// maximum number of concurrent promises to 5
$promise = Promise\each_limit($promiseGenerator(), 5);

// Waiting on an EachPromise will wait on the entire task queue to complete
$promise->wait();
```

promise コルーチン

Guzzle promise ライブラリの強力な機能の 1 つは、従来の同期のワークフローを記述するように、非同期ワークフローを記述できる promise コルーチンを使用できることです。実際、AWS SDK for PHP ではほとんどの高レベルの抽象化で、コルーチン promise を使用します。

複数のバケットを作成し、バケットが利用可能になったときに、バケットにファイルをアップロードする場合、できる限り迅速に完了するようにすべて同時に実行することを考えます。これを行うには、`all()` promise 関数を使用して複数のコルーチン promise を組み合わせます。

```
use GuzzleHttp\Promise;

$uploadFn = function ($bucket) use ($s3Client) {
    return Promise\coroutine(function () use ($bucket, $s3Client) {
        // You can capture the result by yielding inside of parens
        $result = (yield $s3Client->createBucket(['Bucket' => $bucket]));
        // Wait on the bucket to be available
        $waiter = $s3Client->getWaiter('BucketExists', ['Bucket' => $bucket]);
        // Wait until the bucket exists
        yield $waiter->promise();
        // Upload a file to the bucket
        yield $s3Client->putObjectAsync([
            'Bucket' => $bucket,
```

```
            'Key'    => '_placeholder',
            'Body'   => 'Hi!'
        ]);
    });
};

// Create the following buckets
$buckets = ['foo', 'baz', 'bar'];
$promises = [];

// Build an array of promises
foreach ($buckets as $bucket) {
    $promises[] = $uploadFn($bucket);
}

// Aggregate the promises into a single "all" promise
$aggregate = Promise\all($promises);

// You can then() off of this promise or synchronously wait
$aggregate->wait();
```

AWS SDK for PHP バージョン 3 でのハンドラーとミドルウェア

AWS SDK for PHP を拡張するには、主なメカニズムとしてハンドラーとミドルウェアを使用します。各 SDK クライアントクラスには、クライアントの Aws\HandlerList メソッドでアクセスできる独自の getHandlerList() インスタンスがあります。クライアントの HandlerList を取得して変更することによって、クライアントの動作を追加または削除できます。

ハンドラー

ハンドラーは、コマンドおよびリクエストから結果への実際の変換を実行する関数です。ハンドラーは、通常、HTTP リクエストを送信します。ハンドラーをミドルウェアで構成して動作を補強できます。ハンドラー関数は、Aws\CommandInterface と Psr\Http\Message\RequestInterface を受け入れて、Aws\ResultInterface で満たすかまたは Aws\Exception\AwsException 理由で拒否する promise を返します。

次に示しているハンドラーは、呼び出しごとに同じ Mock 結果を返します。

```
use Aws\CommandInterface;
use Aws\Result;
use Psr\Http\Message\RequestInterface;
```

```
use GuzzleHttp\Promise;

$myHandler = function (CommandInterface $cmd, RequestInterface $request) {
    $result = new Result(['foo' => 'bar']);
    return Promise\promise_for($result);
};
```

クライアントのコンストラクタの handler オプションにこのハンドラーを指定することによって、SDK クライアントで使用できます。

```
// Set the handler of the client in the constructor
$s3 = new Aws\S3\S3Client([
    'region'  => 'us-east-1',
    'version' => '2006-03-01',
    'handler' => $myHandler
]);
```

クライアントを作成した後に、setHandler の Aws\ClientInterface メソッドを使用して、そのクライアントのハンドラーを変更することもできます。

```
// Set the handler of the client after it is constructed
$s3->getHandlerList()->setHandler($myHandler);
```

> **Note**
>
> コンストラクトした後、マルチリージョンクライアントのハンドラーを変更するには、Aws\MultiRegionClient の useCustomHandler メソッドを使用します。
>
> ```
> $multiRegionClient->useCustomHandler($myHandler);
> ```

Mock ハンドラー

SDK を使用するテストを記述する場合は MockHandler を使用することをお勧めします。Aws\MockHandler を使用して、模擬結果を返すか、または Mock 例外をスローできます。結果または例外をキューに追加すると、MockHandler は FIFO ルールに従ってそれをキューから削除します。

```
use Aws\Result;
use Aws\MockHandler;
```

```
use Aws\DynamoDb\DynamoDbClient;
use Aws\CommandInterface;
use Psr\Http\Message\RequestInterface;
use Aws\Exception\AwsException;

$mock = new MockHandler();

// Return a mocked result
$mock->append(new Result(['foo' => 'bar']));

// You can provide a function to invoke; here we throw a mock exception
$mock->append(function (CommandInterface $cmd, RequestInterface $req) {
    return new AwsException('Mock exception', $cmd);
});

// Create a client with the mock handler
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'version' => 'latest',
    'handler' => $mock
]);

// Result object response will contain ['foo' => 'bar']
$result = $client->listTables();

// This will throw the exception that was enqueued
$client->listTables();
```

ミドルウェア

ミドルウェアは、コマンドの転送動作を補強して「next」ハンドラーに委任する、特殊なタイプの高レベル関数です。ミドルウェア関数は `Aws\CommandInterface` と `Psr\Http\Message\RequestInterface` を受け入れて、`Aws\ResultInterface` で満たすかまたは `Aws\Exception\AwsException` 理由で拒否する promise を返します。

ミドルウェアは高次関数であり、ミドルウェアを通過したコマンド、リクエスト、または結果が変更されます。ミドルウェアは次のような形式です。

```
use Aws\CommandInterface;
use Psr\Http\Message\RequestInterface;

$middleware = function () {
```

```
    return function (callable $handler) use ($fn) {
        return function (
            CommandInterface $command,
            RequestInterface $request = null
        ) use ($handler, $fn) {
            // Do something before calling the next handler
            // ...
            $promise = $fn($command, $request);
            // Do something in the promise after calling the next handler
            // ...
            return $promise;
        };
    };
};
```

ミドルウェアは、実行する 1 つのコマンドと 1 つのリクエストオブジェクト (省略可能) を受け取ります。ミドルウェアは、リクエストおよびコマンドを補強するかそのままにするかを選択できます。ミドルウェアは、チェーン内の次のハンドラーを呼び出すか、または次のハンドラーを省略して promise を返すことを選択できます。次のハンドラーを呼び出すことによって作成された promise を、その promise の `then` メソッドを使用して、promise をミドルウェアのスタックに戻す前に最終結果またはエラーを変更するように補強できます。

HandlerList

SDK では、コマンドの実行時に使用されるミドルウェアとハンドラーを管理するために `Aws\HandlerList` が使用されます。各 SDK クライアントには独自の `HandlerList` があり、その `HandlerList` のクローンが作成され、クライアントで作成された各コマンドに追加されます。ミドルウェアをクライアントの `HandlerList` に追加することによって、ミドルウェアとデフォルトハンドラーをアタッチして、クライアントで作成された各コマンドで使用されるようにできます。特定のコマンドで所有されている `HandlerList` を変更することによって、そのコマンドに対してミドルウェアを追加および削除できます。

`HandlerList` は、ハンドラーをラップするために使用されるミドルウェアのスタックを表します。ミドルウェアのリストとミドルウェアがハンドラーをラップする順序を管理しやすいように、`HandlerList` では、ミドルウェアスタックは次のように、コマンドの転送のライフサイクルの各部分を表す名前付きのステップに分かれています。

1. `init` - デフォルトパラメーターを追加します
2. `validate` - 必須パラメーターを検証します
3. `build` - HTTP リクエストを送信用にシリアル化します

4. sign - シリアル化した HTTP リクエストに署名します
5. <handler> (ステップではありませんが、実際の転送を実行します)

初期化

このライフサイクルステップは、コマンドの初期化を表し、リクエストはまだシリアル化されていません。このステップは通常、コマンドにデフォルトパラメーターを追加するために使用されます。

init および appendInit メソッドを使用して、prependInit ステップにミドルウェアを追加できます。ミドルウェアは、appendInit では prepend リストの末尾に追加され、prependInit では prepend リストの先頭に追加されます。

```
use Aws\Middleware;

$middleware = Middleware::tap(function ($cmd, $req) {
    // Observe the step
});

// Append to the end of the step with a custom name
$client->getHandlerList()->appendInit($middleware, 'custom-name');
// Prepend to the beginning of the step
$client->getHandlerList()->prependInit($middleware, 'custom-name');
```

validate

このライフサイクルステップは、コマンドの入力パラメーターを検証するために使用されます。

validate および appendValidate メソッドを使用して、prependValidate ステップにミドルウェアを追加できます。ミドルウェアは、appendValidate では validate リストの末尾に追加され、prependValidate では validate リストの先頭に追加されます。

```
use Aws\Middleware;

$middleware = Middleware::tap(function ($cmd, $req) {
    // Observe the step
});

// Append to the end of the step with a custom name
$client->getHandlerList()->appendValidate($middleware, 'custom-name');
// Prepend to the beginning of the step
```

```
$client->getHandlerList()->prependValidate($middleware, 'custom-name');
```

buid (構築)

このライフサイクルステップは、実行するコマンドの HTTP リクエストをシリアル化するために使用されます。下流のライフサイクルイベントは、コマンドと PSR-7 HTTP リクエストを受け取ります。

build および appendBuild メソッドを使用して、prependBuild ステップにミドルウェアを追加できます。ミドルウェアは、appendBuild では build リストの末尾に追加され、prependBuild では build リストの先頭に追加されます。

```
use Aws\Middleware;

$middleware = Middleware::tap(function ($cmd, $req) {
    // Observe the step
});

// Append to the end of the step with a custom name
$client->getHandlerList()->appendBuild($middleware, 'custom-name');
// Prepend to the beginning of the step
$client->getHandlerList()->prependBuild($middleware, 'custom-name');
```

sign

このライフサイクルステップは通常、送信前に HTTP リクエストに署名するために使用されます。通常は、署名エラーを避けるために、署名した後に HTTP リクエストを変更しないでください。

このステップは、HTTP リクエストがハンドラーによって転送される前の、HandlerList での最後のステップです。

sign および appendSign メソッドを使用して、prependSign ステップにミドルウェアを追加できます。ミドルウェアは、appendSign では sign リストの末尾に追加され、prependSign では sign リストの先頭に追加されます。

```
use Aws\Middleware;

$middleware = Middleware::tap(function ($cmd, $req) {
    // Observe the step
});
```

```
// Append to the end of the step with a custom name
$client->getHandlerList()->appendSign($middleware, 'custom-name');
// Prepend to the beginning of the step
$client->getHandlerList()->prependSign($middleware, 'custom-name');
```

使用可能なミドルウェア

この SDK には、クライアントの動作を補強するため、またはクライアントまたはコマンドの実行を確認するために使用できるいくつかのミドルウェアが付属しています。

mapCommand

Aws\Middleware::mapCommand ミドルウェアは、コマンドが HTTP リクエストとしてシリアル化される前に、そのコマンドを変更する必要がある場合に便利です。たとえば、mapCommand を使用して、デフォルトパラメーターの検証または追加を実行できます。mapCommand 関数は、Aws\CommandInterface オブジェクトを受け入れて Aws\CommandInterface オブジェクトを返す呼び出し可能関数を受け入れます。

```
use Aws\Middleware;
use Aws\CommandInterface;

// Here we've omitted the require Bucket parameter. We'll add it in the
// custom middleware.
$command = $s3Client->getCommand('HeadObject', ['Key' => 'test']);

// Apply a custom middleware named "add-param" to the "init" lifecycle step
$command->getHandlerList()->appendInit(
    Middleware::mapCommand(function (CommandInterface $command) {
        $command['Bucket'] = 'mybucket';
        // Be sure to return the command!
        return $command;
    }),
    'add-param'
);
```

mapRequest

Aws\Middleware::mapRequest ミドルウェアは、コマンドが HTTP リクエストとしてシリアル化された後、かつ送信される前に、リクエストを変更する必要がある場合に便利です。たとえば、このミドルウェアを使用して、リクエストにカスタム HTTP ヘッダーを追加できます。mapRequest

関数は、Psr\Http\Message\RequestInterface 引数を受け入れて Psr\Http\Message\RequestInterface オブジェクトを返す呼び出し可能関数を受け入れます。

```
use Aws\Middleware;
use Psr\Http\Message\RequestInterface;

// Create a command so that we can access the handler list
$command = $s3Client->getCommand('HeadObject', [
    'Key'    => 'test',
    'Bucket' => 'mybucket'
]);

// Apply a custom middleware named "add-header" to the "build" lifecycle step
$command->getHandlerList()->appendBuild(
    Middleware::mapRequest(function (RequestInterface $request) {
        // Return a new request with the added header
        return $request->withHeader('X-Foo-Baz', 'Bar');
    }),
    'add-header'
);
```

これで、コマンドが実行されたときに、カスタムヘッダー付きで送信されるようになりました。

> **Important**
>
> このミドルウェアは、build ステップの最後でハンドラーリストに追加されています。これは、リクエストが作成された後に、このミドルウェアが呼び出されるようにするためです。

mapResult

Aws\Middleware::mapResult ミドルウェアは、コマンドの実行結果を変更する必要がある場合に便利です。mapResult 関数は、Aws\ResultInterface 引数を受け入れて Aws\ResultInterface オブジェクトを返す呼び出し可能関数を受け入れます。

```
use Aws\Middleware;
use Aws\ResultInterface;

$command = $s3Client->getCommand('HeadObject', [
    'Key'    => 'test',
    'Bucket' => 'mybucket'
```

```
]);

$command->getHandlerList()->appendSign(
    Middleware::mapResult(function (ResultInterface $result) {
        // Add a custom value to the result
        $result['foo'] = 'bar';
        return $result;
    })
);
```

これで、コマンドが実行されたときに、返される結果に foo 属性が含まれるようになりました。

history

history ミドルウェアは、SDK で、期待どおりのコマンドが実行され、期待どおりの HTTP リクエストが送信され、期待どおりの結果が受け取られたかどうかをテストする場合に便利です。これは、実質的には、ウェブブラウザの履歴と同じように動作するミドルウェアです。

```
use Aws\History;
use Aws\Middleware;

$ddb = new Aws\DynamoDb\DynamoDbClient([
    'version' => 'latest',
    'region'  => 'us-west-2'
]);

// Create a history container to store the history data
$history = new History();

// Add the history middleware that uses the history container
$ddb->getHandlerList()->appendSign(Middleware::history($history));
```

Aws\History 履歴コンテナには、デフォルトでは 10 個のエントリが保存され、それを越えるエントリは消去されます。このエントリ数は、保持するエントリ数をコンストラクタに渡すことによってカスタマイズできます。

```
// Create a history container that stores 20 entries
$history = new History(20);
```

history ミドルウェアに渡すリクエストの実行後に、履歴コンテナを調べることができます。

```
// The object is countable, returning the number of entries in the container
```

```
count($history);

// The object is iterable, yielding each entry in the container
foreach ($history as $entry) {
    // You can access the command that was executed
    var_dump($entry['command']);
    // The request that was serialized and sent
    var_dump($entry['request']);
    // The result that was received (if successful)
    var_dump($entry['result']);
    // The exception that was received (if a failure occurred)
    var_dump($entry['exception']);
}

// You can get the last Aws\CommandInterface that was executed. This method
// will throw an exception if no commands have been executed.
$command = $history->getLastCommand();

// You can get the last request that was serialized. This method will throw an
 exception
// if no requests have been serialized.
$request = $history->getLastRequest();

// You can get the last return value (an Aws\ResultInterface or Exception).
// The method will throw an exception if no value has been returned for the last
// executed operation (e.g., an async request has not completed).
$result = $history->getLastReturn();

// You can clear out the entries using clear
$history->clear();
```

tap

tap ミドルウェアはオブザーバーとして使用されます。このミドルウェアを使用して、ミドルウェアのチェーンを通じてコマンドを送信するときに関数を呼び出すことができます。この tap 関数は、実行される Aws\CommandInterface および Psr\Http\Message\RequestInterface (省略可能) を受け入れる呼び出し可能関数を受け入れます。

```
use Aws\Middleware;

$s3 = new Aws\S3\S3Client([
    'region'  => 'us-east-1',
    'version' => '2006-03-01'
```

```
]);

$handlerList = $s3->getHandlerList();

// Create a tap middleware that observes the command at a specific step
$handlerList->appendInit(
    Middleware::tap(function (CommandInterface $cmd, RequestInterface $req = null) {
        echo 'About to send: ' . $cmd->getName() . "\n";
        if ($req) {
            echo 'HTTP method: ' . $request->getMethod() . "\n";
        }
    }
);
```

カスタムハンドラーの作成

ハンドラーは、`Aws\CommandInterface` と `Psr\Http\Message\RequestInterface` を受け入れて、`GuzzleHttp\Promise\PromiseInterface` で満たすかまたは `Aws\ResultInterface` 理由で拒否する `Aws\Exception\AwsException` を返すだけの関数です。

SDK にはいくつかの `@http` オプションがありますが、以下のオプションの使用法を知る必要があるのはハンドラーだけです。

- connect_timeout
- debug
- decode_content (省略可能)
- delay
- progress (省略可能)
- proxy
- sink
- synchronous (省略可能)
- stream (省略可能)
- timeout
- verify
- http_stats_receiver (省略可能) - stats 設定パラメーターを使用してリクエストされた場合に、HTTP 転送の統計情報の連想配列を付けて呼び出すための関数。

このオプションが省略可能として指定されている場合を除いて、ハンドラーはこのオプションを処理できる必要があり、そうでない場合は拒否された promise を返す必要があります。

特定の `@http` オプションを処理することに加えて、ハンドラーは次の形式の `User-Agent` ヘッダーを追加する必要があります。ここで、「3.X」は `Aws\Sdk::VERSION` に置き換えることができ、「HandlerSpecificData/version ...」はハンドラー固有の User-Agent 文字列に置き換える必要があります。

```
User-Agent: aws-sdk-php/3.X HandlerSpecificData/version ...
```

AWS SDK for PHP バージョン 3 でのストリーム

PSR-7 HTTP メッセージ標準の統合の一環として、AWS SDK for PHP では PSR-7 StreamInterface を PHP ストリームの抽象化として内部的に使用しています。blob として定義されている入力フィールドを持つすべてのコマンド (BodyS3::PutObject コマンドの パラメーターなど) では、文字列、PHP ストリームリソース、または `Psr\Http\Message\StreamInterface` のインスタンスを指定できます。

> **Warning**
>
> SDK は、コマンドへの入力パラメーターとして指定されたすべての RAW PHP ストリームリソースの所有権を持ちます。そのストリームはユーザーに代わって処理され、閉じられます。
> SDK のオペレーションとユーザーのコードの間でストリームを共有する必要がある場合は、そのストリームをコマンドパラメーターとして含める前に、ストリームを `GuzzleHttp\Psr7\Stream` のインスタンス内にラップします。ストリームは SDK で処理されるため、ユーザーのコードではストリームの内部カーソルの移動を把握する必要があります。Guzzle ストリームでは、ストリームが PHP のガベージコレクターによって破棄されるときに、基になるストリームリソースの `fclose` が呼び出されるため、ユーザーがストリームを閉じる必要はありません。

ストリームデコレータ

Guzzle には、いくつかのストリームデコレータが付属していて、それを使用すると、SDK と Guzzle が、コマンドへの入力パラメーターとして指定されたストリーミングリソースとどのようにやり取りするかを制御できます。これらのデコレータでは、指定されたストリームをハンドラーがど

のように読み取りおよびシークできるかを変更できます。以下に部分的なリストを示していますが、完全なリストは GuzzleHttpPsr7 repository にあります。

AppendStream

GuzzleHttp\Psr7\AppendStream

複数のストリームから順々に読み取ります。

```
use GuzzleHttp\Psr7;

$a = Psr7\stream_for('abc, ');
$b = Psr7\stream_for('123.');
$composed = new Psr7\AppendStream([$a, $b]);

$composed->addStream(Psr7\stream_for(' Above all listen to me'));

echo $composed(); // abc, 123. Above all listen to me.
```

CachingStream

GuzzleHttp\Psr7\CachingStream

シーク不可のストリームで、以前に読み取ったバイトにシークできるようにするために使用されます。これは、ストリームの巻き戻しが必要であるために、シーク不可のエンティティ本文の転送が失敗する場合に便利です (たとえば、リダイレクトによるストリーム)。リモートストリームから読み取られるデータは PHP の一時ストリームにバッファされるため、以前に読み取られたバイトがメモリにキャッシュされ、その後にディスクに書き込まれます。

```
use GuzzleHttp\Psr7;

$original = Psr7\stream_for(fopen('http://www.google.com', 'r'));
$stream = new Psr7\CachingStream($original);

$stream->read(1024);
echo $stream->tell();
// 1024

$stream->seek(0);
echo $stream->tell();
// 0
```

InflateStream

GuzzleHttp\Psr7\InflateStream

PHP の zlib.inflate フィルタを使用して、gzipped コンテンツを解凍または圧縮します。

このストリームデコレータは、指定されたストリームの最初の 10 バイトをスキップして gzip ヘッダーを除去し、ストリームを PHP ストリームリソースに変換してから、zlib.inflate フィルタを追加します。そのストリームは、Guzzle ストリームリソースに変換し直されて、Guzzle ストリームとして使用されます。

LazyOpenStream

GuzzleHttp\Psr7\LazyOpenStream

ストリームに対する I/O オペレーションが実行された後にのみ開かれるファイルに対する読み取りまたは書き込みを遅延します。

```
use GuzzleHttp\Psr7;

$stream = new Psr7\LazyOpenStream('/path/to/file', 'r');
// The file has not yet been opened...

echo $stream->read(10);
// The file is opened and read from only when needed.
```

LimitStream

GuzzleHttp\Psr7\LimitStream

既存のストリームオブジェクトのサブセットまたはスライスを読み取るために使用されます。これは、サイズが大きいファイルを小さく分割して、チャンクに分けて送信する場合に便利です (例: Amazon S3 マルチパートアップロード API)。

```
use GuzzleHttp\Psr7;

$original = Psr7\stream_for(fopen('/tmp/test.txt', 'r+'));
echo $original->getSize();
// >>> 1048576

// Limit the size of the body to 1024 bytes and start reading from byte 2048
```

```
$stream = new Psr7\LimitStream($original, 1024, 2048);
echo $stream->getSize();
// >>> 1024
echo $stream->tell();
// >>> 0
```

NoSeekStream

GuzzleHttp\Psr7\NoSeekStream

ストリームをラップして、シークを許可しません。

```
use GuzzleHttp\Psr7;

$original = Psr7\stream_for('foo');
$noSeek = new Psr7\NoSeekStream($original);

echo $noSeek->read(3);
// foo
var_export($noSeek->isSeekable());
// false
$noSeek->seek(0);
var_export($noSeek->read(3));
// NULL
```

PumpStream

GuzzleHttp\Psr7\PumpStream

PHP 呼び出し可能関数からデータを取り出す読み取り専用ストリームを提供します。

指定された呼び出し可能関数を呼び出されたときに、PumpStream は読み取りをリクエストされたデータ量を呼び出し可能関数に渡します。呼び出し可能関数は、その値を無視することも、リクエストされたバイト数と異なる量のデータを返すこともできます。呼び出し可能関数から返された余分なデータは、PumpStream の read() 関数を使用してドレインされるまで、内部的にバッファに入れられます。呼び出し可能関数は、読み取るデータがなくなった場合に false を返す必要があります。

ストリームデコレータの実装

GuzzleHttp\Psr7\StreamDecoratorTrait を使用すると、ストリームデコレータノードの作成は非常に簡単です。このトレイトでは、基になるストリームにプロキシすることによって

Psr\Http\Message\StreamInterface を実装するメソッドが提供されています。use を StreamDecoratorTrait して、カスタムメソッドを実装するだけです。

たとえば、ストリームから最後のバイトが読み取られるたびに特定の関数を呼び出すことを考えてみます。そうするには、read() メソッドをオーバーライドことで実装できます。

```
use Psr\Http\Message\StreamInterface;
use GuzzleHttp\Psr7\StreamDecoratorTrait;

class EofCallbackStream implements StreamInterface
{
    use StreamDecoratorTrait;

    private $callback;

    public function __construct(StreamInterface $stream, callable $cb)
    {
        $this->stream = $stream;
        $this->callback = $cb;
    }

    public function read($length)
    {
        $result = $this->stream->read($length);

        // Invoke the callback when EOF is hit
        if ($this->eof()) {
            call_user_func($this->callback);
        }

        return $result;
    }
}
```

このデコレータを、任意の既存ストリームに追加して、このように使用できます。

```
use GuzzleHttp\Psr7;

$original = Psr7\stream_for('foo');

$eofStream = new EofCallbackStream($original, function () {
    echo 'EOF!';
});
```

```
$eofStream->read(2);
$eofStream->read(1);
// echoes "EOF!"
$eofStream->seek(0);
$eofStream->read(3);
// echoes "EOF!"
```

AWS SDK for PHP バージョン 3 でのページネーター

一部の AWS サービスオペレーションではページ分割され、結果が一部切り捨てられて応答します。例えば、Amazon S3 の ListObjects オペレーションが一度に返すことができるオブジェクトは最大で 1,000 個です。このようなオペレーション (通常は名前が「list」や「describe」で始まる) では、結果セット全体を取得するために、後続のリクエストでトークン (マーカー) パラメータを指定する必要があります。

ページネーターは AWS SDK for PHP の機能で、開発者がページ分割 API を簡単に使用できるようにこのプロセスを抽象化します。ページネーターは、実質的には結果のイテレーターです。ページネーターは、クライアントの getPaginator( ) メソッドを使用して作成されます。getPaginator( ) を呼び出す際に、オペレーションの名前とオペレーションの引数を (オペレーションを実行する場合と同じ方法で) 指定する必要があります。foreach を使用してページネーターオブジェクトを反復処理して、個々の Aws\Result オブジェクトを取得します。

```
$results = $s3Client->getPaginator('ListObjects', [
    'Bucket' => 'my-bucket'
]);

foreach ($results as $result) {
    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
}
```

ページネーターオブジェクト

getPaginator( ) メソッドによって返されるオブジェクトは、Aws\ResultPaginator クラスのインスタンスです。このクラスでは、PHP のネイティブの iterator インターフェイスが実装されているため、foreach で機能します。また、iterator_to_array などのイテレーター関数で使用したり、 オブジェクトなどの SPL イテレーターLimitIteratorと組み合わせたりできます。

ページネーターオブジェクトは、結果の「ページ」を一度に 1 ページのみ保持し、遅延して実行されます。つまり、ページネーターは、結果の現在のページを生成するのに必要な数のリクエストを作成します。例えば、Amazon S3 の `ListObjects` オペレーションは一度に最大 1,000 個のオブジェクトのみを返すため、バケットに 10,000 個以下のオブジェクトがある場合、ページネーターは合計 10 回のリクエストを行う必要があります。結果を反復処理する場合、反復処理を開始したときに最初のリクエストが実行され、ループの 2 回目の反復で 2 つ目のリクエストが実行されます。

結果からのデータの列挙

ページネーターオブジェクトには、結果の 1 つのセット内のデータに対するイテレーターを作成できる `search()` という名前のメソッドがあります。`search()` を呼び出す際に、抽出するデータを指定するための JMESPath 式を指定します。`search()` を呼び出すと、結果の各ページに対する式の結果を生成するイテレーターが返されます。これは、返されたイテレーターを反復処理するときに、遅延して評価されます。

次の例は、前のコード例と同等ですが、より簡潔にするために `ResultPaginator::search()` メソッドを使用しています。

```
$results = $s3Client->getPaginator('ListObjects', [
    'Bucket' => 'my-bucket'
]);

foreach ($results->search('Contents[].Key') as $key) {
    echo $key . "\n";
}
```

JMESPath 式を使用すると、かなり複雑なことを行うことができます。たとえば、すべてのオブジェクトキーおよび共通のプレフィックス (つまり、バケットの `ls`) を出力する場合は、次のようにします。

```
// List all prefixes ("directories") and objects ("files") in the bucket
$results = $s3Client->getPaginator('ListObjects', [
    'Bucket'    => 'my-bucket',
    'Delimiter' => '/'
]);

$expression = '[CommonPrefixes[].Prefix, Contents[].Key][]';
foreach ($results->search($expression) as $item) {
    echo $item . "\n";
}
```

非同期ページ割り

each() の Aws\ResultPaginator メソッドのコールバックを指定することによって、ページネーターの結果を非同期に反復処理できます。そのコールバックは、ページネーターで生成される値ごとに呼び出されます。

```
$results = $s3Client->getPaginator('ListObjects', [
    'Bucket' => 'my-bucket'
]);

$promise = $results->each(function ($result) {
    echo 'Got ' . var_export($result, true) . "\n";
});
```

> **Note**
>
> each() メソッドを使用すると、同時に他のリクエストを非同期に送信しながら、API オペレーションの結果をページ分割できます。

そのコールバックからの Null 以外の戻り値は、基になるコルーチンベースの promise によって生成されます。つまり、残っている項目の反復処理を続行する (実質的には他の promise を反復にマージして) 前に、解決する必要があるコールバックから promise を返すことができます。コールバックによって返された直前の Null 以外の値は、promise を任意のダウンストリーム promise に対して満たした結果です。直前の戻り値が promise である場合、その promise の解決はダウンストリーム promise を満たすかまたは拒否した結果です。

```
// Delete all keys that end with "Foo"
$promise = $results->each(function ($result) use ($s3Client) {
    if (substr($result['Key'], -3) === 'Foo') {
        // Merge this promise into the iterator
        return $s3Client->deleteAsync([
            'Bucket' => 'my-bucket',
            'Key'    => 'Foo'
        ]);
    }
});

$promise
    ->then(function ($result) {
```

```
        // Result would be the last result to the deleteAsync operation
    })
    ->otherwise(function ($reason) {
        // Reason would be an exception that was encountered either in the
        // call to deleteAsync or calls performed while iterating
    });

// Forcing a synchronous wait will also wait on all of the deleteAsync calls
$promise->wait();
```

AWS SDK for PHP バージョン 3 でのウェーター

ウェーターを使用すると、リソースをポーリングしてリソースが特定の状態になるまで待機するための抽象化された方法を提供することによって、結果整合性のあるシステムでの操作が容易になります。クライアントでサポートされているウェーターの一覧は、サービスクライアントの単一のバージョンの API ドキュメントに記載されています。そこに移動するには、API ドキュメントのクライアントのページに移動し、特定のバージョン番号 (日付で表される) に移動し、[Waiters] (ウェーター) セクションまでスクロールします。このリンクをクリックすると、S3 のウェーターセクションに移動します。

次の例では、Amazon S3 クライアントを使用してバケットを作成した後に、ウェーターを使用して、そのバケットが利用可能になるまで待機しています。

```
// Create a bucket
$s3Client->createBucket(['Bucket' => 'my-bucket']);

// Wait until the created bucket is available
$s3Client->waitUntil('BucketExists', ['Bucket' => 'my-bucket']);
```

バケットに対するウェーターのポーリング回数が多すぎる場合は、`\RuntimeException` 例外がスローされます。

ウェーターの設定

ウェーターは、設定オプションの連想配列によって駆動されます。特定のウェーターで使用されるすべてのオプションにはデフォルト値がありますが、それをオーバーライドして異なる待機方法を指定できます。

ウェーターの設定オプションは、`@waiter` オプションの連想配列をクライアントの `$args` および `waitUntil()` の `getWaiter()` 引数に渡すことによって変更できます。

```
// Providing custom waiter configuration options to a waiter
$s3Client->waitUntil('BucketExists', [
    'Bucket'  => 'my-bucket',
    '@waiter' => [
        'delay'       => 3,
        'maxAttempts' => 10
    ]
]);
```

delay (int)

ポーリング試行間隔の秒数。各ウェーターにはデフォルトの delay 設定値がありますが、特定のユースケースに合わせてその設定を変更できます。

maxAttempts (int)

発行するポーリング試行の最大回数。この回数を超えるとウェーターは失敗します。このオプションにより、リソースを無期限に待機しなくなります。各ウェーターにはデフォルトの maxAttempts 設定値がありますが、特定のユースケースに合わせてその設定を変更できます。

initDelay (int)

最初のポーリング試行までの待機時間 (秒)。これは、目的の状態になるまでにしばらく時間がかかることがわかっているリソースを待機する場合に便利です。

before (callable)

各ポーリング試行の前に呼び出す、PHP で呼び出し可能な関数。この呼び出し可能な関数の呼び出しでは、実行されようとしている Aws\CommandInterface コマンドと実行済みの試行回数が渡されます。before 呼び出し可能関数を使用すると、実行する前にコマンドを変更したり、進行状況を提供したりできます。

```
use Aws\CommandInterface;

$s3Client->waitUntil('BucketExists', [
    'Bucket'  => 'my-bucket',
    '@waiter' => [
        'before' => function (CommandInterface $command, $attempts) {
            printf(
                "About to send %s. Attempt %d\n",
                $command->getName(),
                $attempts
            );
        }
```

```
    ]
]);
```

非同期の待機

同期的に待機するだけでなく、他のリクエストの送信中や複数のリソースの同時待機中に、ウェーターを呼び出して非同期で待機することもできます。

クライアントの `getWaiter($name, array $args = [])` メソッドを使用してクライアントからウェーターを取得することによって、ウェーターの promise にアクセスできます。ウェーターを開始するには、ウェーターの `promise()` メソッドを使用します。ウェーターの promise には、ウェーターで実行された最後の `Aws\CommandInterface` が満たされ、エラー発生時には `RuntimeException` で拒否されます。

```
use Aws\CommandInterface;

$waiterName = 'BucketExists';
$waiterOptions = ['Bucket' => 'my-bucket'];

// Create a waiter promise
$waiter = $s3Client->getWaiter($waiterName, $waiterOptions);

// Initiate the waiter and retrieve a promise
$promise = $waiter->promise();

// Call methods when the promise is resolved.
$promise
    ->then(function () {
        echo "Waiter completed\n";
    })
    ->otherwise(function (\Exception $e) {
        echo "Waiter failed: " . $e . "\n";
    });

// Block until the waiter completes or fails. Note that this might throw
// a \RuntimeException if the waiter fails.
$promise->wait();
```

promise ベースのウェーター API を公開すると、ある程度強力でオーバーヘッドが比較的小さいユースケースに効果があります。たとえば、複数のリソースを待機する場合に、最初に解決されたウェーターで何か行うにはどうしたらよいでしょうか。

```
use Aws\CommandInterface;

// Create an array of waiter promises
$promises = [
    $s3Client->getWaiter('BucketExists', ['Bucket' => 'a'])->promise(),
    $s3Client->getWaiter('BucketExists', ['Bucket' => 'b'])->promise(),
    $s3Client->getWaiter('BucketExists', ['Bucket' => 'c'])->promise()
];

// Initiate a race between the waiters, fulfilling the promise with the
// first waiter to complete (or the first bucket to become available)
$any = Promise\any($promises)
    ->then(function (CommandInterface $command) {
        // This is invoked with the command that succeeded in polling the
        // resource. Here we can know which bucket won the race.
        echo "The {$command['Bucket']} waiter completed first!\n";
    });

// Force the promise to complete
$any->wait();
```

AWS SDK for PHP バージョン 3 での JMESPath 式

JMESPath により JSON ドキュメントから要素を抽出する方法を宣言により指定できます。AWS SDK for PHP は AWS SDK for PHP バージョン 3 でのページネーターや AWS SDK for PHP バージョン 3 でのウェーターなど、一部の高レベルの抽象化を行うために jmespath.php に依存しますが、`Aws\ResultInterface` や `Aws\ResultPaginator` を検索する JMESPath も公開します。

オンラインの JMESPath の例を試すことにより、ブラウザで JMESPath を実際に操作してみることができます。言語の詳細 (使用可能な式と関数など) については、JMESPath の仕様を参照してください。

AWS CLI は JMESPath をサポートしています。CLI 出力用に記述した式は、AWS SDK for PHP 用に記述した式と 100 パーセントの互換性があります。

結果からのデータの抽出

`Aws\ResultInterface` インターフェイスには、JMESPath 式に基づいて結果モデルからデータを抽出する `search($expression)` メソッドがあります。結果オブジェクトからデータをクエリする JMESPath 式を使用すると、共通条件付きコードを削除し、抽出対象データを簡潔に表現できます。

この機能を示すため、最初に以下のデフォルトの JSON 出力で開始します。この出力では、2 つの Amazon Elastic Block Store (Amazon EBS) ボリュームが個別の Amazon EC2 インスタンスにアタッチされています。

```
$result = $ec2Client->describeVolumes();
// Output the result data as JSON (just so we can clearly visualize it)
echo json_encode($result->toArray(), JSON_PRETTY_PRINT);
```

```
{
    "Volumes": [
        {
            "AvailabilityZone": "us-west-2a",
            "Attachments": [
                {
                    "AttachTime": "2013-09-17T00:55:03.000Z",
                    "InstanceId": "i-a071c394",
                    "VolumeId": "vol-e11a5288",
                    "State": "attached",
                    "DeleteOnTermination": true,
                    "Device": "/dev/sda1"
                }
            ],
            "VolumeType": "standard",
            "VolumeId": "vol-e11a5288",
            "State": "in-use",
            "SnapshotId": "snap-f23ec1c8",
            "CreateTime": "2013-09-17T00:55:03.000Z",
            "Size": 30
        },
        {
            "AvailabilityZone": "us-west-2a",
            "Attachments": [
                {
                    "AttachTime": "2013-09-18T20:26:16.000Z",
                    "InstanceId": "i-4b41a37c",
                    "VolumeId": "vol-2e410a47",
                    "State": "attached",
                    "DeleteOnTermination": true,
                    "Device": "/dev/sda1"
                }
            ],
            "VolumeType": "standard",
            "VolumeId": "vol-2e410a47",
```

```
            "State": "in-use",
            "SnapshotId": "snap-708e8348",
            "CreateTime": "2013-09-18T20:26:15.000Z",
            "Size": 8
        }
    ],
    "@metadata": {
        "statusCode": 200,
        "effectiveUri": "https:\/\/ec2.us-west-2.amazonaws.com",
        "headers": {
            "content-type": "text\/xml;charset=UTF-8",
            "transfer-encoding": "chunked",
            "vary": "Accept-Encoding",
            "date": "Wed, 06 May 2015 18:01:14 GMT",
            "server": "AmazonEC2"
        }
    }
}
```

最初に、次のコマンドで、ボリュームリストの最初のボリュームのみを取得できます。

```
$firstVolume = $result->search('Volumes[0]');
```

ここで、wildcard-index 式 [*] を使用して、リスト全体を反復的に処理し、3 つの要素を抽出します。 VolumeId は ID に、AvailabilityZone は AZ に変更され、Size は Size のままです。multi-hash 式を wildcard-index 式の後ろに配置してこれらの要素を抽出して名前を変更できます。

```
$data = $result->search('Volumes[*].{ID: VolumeId, AZ: AvailabilityZone, Size: Size}');
```

これにより、PHP データの配列は次のようになります。

```
array(2) {
  [0] =>
  array(3) {
    'AZ' =>
    string(10) "us-west-2a"
    'ID' =>
    string(12) "vol-e11a5288"
    'Size' =>
    int(30)
  }
```

```
  [1] =>
  array(3) {
    'AZ' =>
    string(10) "us-west-2a"
    'ID' =>
    string(12) "vol-2e410a47"
    'Size' =>
    int(8)
  }
}
```

multi-hash 表記では、key1.key2[0].key3 など連鎖キーを使用して、構造内で深く入れ子になった要素を抽出することもできます。以下の例では、単純に Attachments[0].InstanceId に対して InstanceId キーのエイリアスを作成して、これを示します。(ほとんどの場合、JMESPath 式は空白を無視します)。

```
$expr = 'Volumes[*].{ID: VolumeId,
                     InstanceId: Attachments[0].InstanceId,
                     AZ: AvailabilityZone,
                     Size: Size}';

$data = $result->search($expr);
var_dump($data);
```

前の式では、次のデータを出力します。

```
array(2) {
  [0] =>
  array(4) {
    'ID' =>
    string(12) "vol-e11a5288"
    'InstanceId' =>
    string(10) "i-a071c394"
    'AZ' =>
    string(10) "us-west-2a"
    'Size' =>
    int(30)
  }
  [1] =>
  array(4) {
    'ID' =>
    string(12) "vol-2e410a47"
```

```
    'InstanceId' =>
    string(10) "i-4b41a37c"
    'AZ' =>
    string(10) "us-west-2a"
    'Size' =>
    int(8)
  }
}
```

multi-list 式: [key1, key2] で複数の要素をフィルタリングすることもできます。これにより、フィルタリングされたすべての属性が、型に関係なく、オブジェクトごとに 1 つの順序付きリスト形式になります。

```
$expr = 'Volumes[*].[VolumeId, Attachments[0].InstanceId, AvailabilityZone, Size]';
$data = $result->search($expr);
var_dump($data);
```

前の検索を実行すると、次のデータが生成されます。

```
array(2) {
  [0] =>
  array(4) {
    [0] =>
    string(12) "vol-e11a5288"
    [1] =>
    string(10) "i-a071c394"
    [2] =>
    string(10) "us-west-2a"
    [3] =>
    int(30)
  }
  [1] =>
  array(4) {
    [0] =>
    string(12) "vol-2e410a47"
    [1] =>
    string(10) "i-4b41a37c"
    [2] =>
    string(10) "us-west-2a"
    [3] =>
    int(8)
  }
```

```
}
```

特定のフィールドの値によって結果をフィルタリングする、filter 式を使用します。次のクエリの例では、us-west-2a アベイラビリティーゾーンのボリュームのみを出力します。

```
$data = $result->search("Volumes[?AvailabilityZone ## 'us-west-2a']");
```

JMESPath は関数式もサポートしています。例えば、上記と同じクエリを実行する場合、すべてのボリュームを取得する代わりに、「us-」で始まる AWS リージョン内のボリュームを取得します。次の式では、starts_with 関数を使用して、文字列リテラル us- を渡します。この関数の結果は JSON のリテラル値 true と比較されます。フィルタ処理を通じて true を返したフィルタ述語の結果のみを渡します。

```
$data = $result->search('Volumes[?starts_with(AvailabilityZone, 'us-') ## `true`]');
```

ページネーターからのデータの抽出

AWS SDK for PHP バージョン 3 でのページネーターガイドにあるように、Aws\ResultPaginator オブジェクトではページング可能な API オペレーションの結果を出力します。AWS SDK for PHP により Aws\ResultPaginator オブジェクトからフィルタリングされたデータを抽出し、それを繰り返すことができます。特に JMESPath 式の結果がマップ関数であるイテレーターに flat-map を実装します。

たとえば、1 MB より大きなバケットにあるオブジェクトのみを出力する iterator を作成する場合です。これを実現するには、最初に ListObjects ページネーターを作成し、search() 関数をページネーターに適用します。さらにページ分割されたデータにフラットマップイテレーターを作成します。

```
$result = $s3Client->getPaginator('ListObjects', ['Bucket' => 't1234']);
$filtered = $result->search('Contents[?Size > `1048576`]');

// The result yielded as $data will be each individual match from
// Contents in which the Size attribute is > 1048576
foreach ($filtered as $data) {
    var_dump($data);
}
```

AWS Common Runtime (AWS CRT) 拡張機能を使用する

AWS CRT ライブラリは、いくつかの AWS SDKsに対して、パフォーマンスが高く、フットプリントが最小限である基本的な機能を提供します。このトピックでは、SDK for PHP AWS で CRT が使用されるタイミングと、CRT AWS 拡張機能のインストール方法について説明します。

CRT AWS 拡張機能をインストールする必要がある場合

SDK for PHP は、CRT AWS ライブラリの認証およびチェックサム機能を使用します。AWS CRT 拡張機能は、以下を使用する場合に必要です。

- Amazon S3 マルチリージョンアクセスポイント
- Amazon EventBridge グローバルエンドポイント
- Amazon Simple Storage Service (Amazon S3) の CRC-32C チェックサムアルゴリズムAmazon S3

上記の機能を使用していて、CRT 拡張が PHP AWS 環境にインストールされていない場合、SDK for PHP はエラーメッセージを報告し、拡張機能をインストールするように促します。

Common Runtime (AWSAWS CRT) 拡張機能をインストールする

CRT AWS 拡張機能をインストールする方法については、 のGitHubリポジトリ aws-crt-phpのメインページを参照してください。

AWS SDK for PHP のバージョン 2 からのアップグレード

このトピックでは、コードを AWS SDK for PHP バージョン 3 に移行する方法、および新しいバージョンと SDK バージョン 2 の相違点について説明します。

> **Note**
>
> SDK の基本的な使用パターン (例: `$result = $client->operation($params);`) はバージョン 2 からバージョン 3 で変更されていないため、スムーズに移行できます。

序章

AWS SDK for PHP バージョン 3 では、SDK の機能を改善するためのかなりの労力のたまものであり、2 年以上にわたるカスタマーフィードバックの取り込み、依存関係のアップグレード、パフォーマンスの改善、最新の PHP 標準の採用を行っています。

バージョン 3 の新機能

AWS SDK for PHP バージョン 3 は PSR-4 標準と PSR-7 標準に従っていて、将来は SemVer 標準に従う予定です。

その他の新機能は次のとおりです。

- サービスクライアントの動作をカスタマイズするためのミドルウェアシステム
- ページ分割された結果を反復処理するための柔軟なページネーター
- JMESPath を使用して結果オブジェクトとページネーターオブジェクトからデータをクエリする機能
- `'debug'` 設定オプションによる簡単なデバッグ

分離された HTTP レイヤー

- デフォルトではリクエストの送信に Guzzle 6 が使用されますが、Guzzle 5 もサポートされています。
- SDK は、cURL を利用できない環境でも機能します。
- カスタム HTTP ハンドラーもサポートされています。

非同期リクエスト

- ウェーターやマルチパートアップローダーなどの機能は非同期でも使用できます。
- promise およびコルーチン を使用して非同期ワークフローを作成できます。
- 同時処理やバッチ処理されるリクエストのパフォーマンスが向上しました。

バージョン 2 との相違点

プロジェクトの依存関係が更新されました

このバージョンで SDK の依存関係が変更されています。

- SDK で PHP 5.5 以降が必要になりました。SDK コード内で ジェネレーターを積極的に使用しています。
- Guzzle 6 (または 5) を使用するように SDK をアップグレードしました。これにより、リクエストを AWS のサービスに送信するために SDK で使用される、基になる HTTP クライアントの実装が提供されます。最新バージョンの Guzzle が付属し、非同期リクエスト、スワップ可能な HTTP ハンドラー、PSR-7 準拠、パフォーマンスの向上など、いくつかの改善が行われました。
- PHP-FIG (`psr/http-message`) による PSR-7 パッケージでは、HTTP リクエスト、HTTP レスポンス、URL、およびストリームを表すためのインターフェイスが定義されています。これらのインターフェイスは SDK と Guzzle 全体で使用されているため、他の PSR-7 準拠パッケージとの相互運用性が提供されています。
- Guzzle の PSR-7 実装 (`guzzlehttp/psr7`) では、PSR-7 のインターフェイスいくつかの便利なクラスと関数の実装が提供されています。SDK と Guzzle 6 はどちらも、このパッケージに大きく依存しています。
- Guzzle の Promises/A+ 実装 (`guzzlehttp/promises`) は、非同期のリクエストおよびコルーチンを管理するインターフェイスを提供するために、SDK と Guzzle 全体で使用されています。Guzzle のマルチ cURL HTTP ハンドラーでは、非同期リクエストを実現するために最終的にノンブロッキング I/O モデルが実装されていますが、このパッケージではそのパラダイム内でプログラムに機能を提供しています。詳細については、「AWS SDK for PHP バージョン 3 での promise」を参照してください。
- この SDK では、`Aws\Result::search()` および `Aws\ResultPaginator::search()` メソッドでのデータクエリ機能を提供するために、JMESPath の PHP 実装 (`mtdowling/jmespath.php`) が使用されています。詳細については、「AWS SDK for PHP バージョン 3 での JMESPath 式」を参照してください。

リージョンオプションとバージョンオプションは必須オプションになりました

任意のサービスに対してクライアントをインスタンス化する際に、`'region'` オプションと `'version'` オプションを指定します。AWS SDK for PHP バージョン 2 では、`'version'` は完全に省略可能であり、`'region'` は場合によって省略可能でした。バージョン 3 では、どちらも常に必要です。これらのオプションの両方を明示的に指定することによって、コーディングしている API

バージョンと AWS リージョンに縛りつけることができます。新しい API バージョンが作成された場合や新しい AWS リージョンが利用可能になった場合に、ユーザーが設定を明示的に更新する準備が整うまでは、互換性を破る可能性がある変更の影響を受けなくなります。

> **Note**
>
> どの API バージョンを使用しているか気にならない場合は、`'version'` オプションを `'latest'` に設定するだけで済みます。ただし、本稼働用のコードでは API バージョン番号を明示的に設定することをお勧めします。
> すべての AWS リージョンですべてのサービスを利用できるわけではありません。利用可能なリージョンの一覧については、「リージョンとエンドポイント」を参照してください。
> 単一のグローバルエンドポイント (Amazon Route 53、AWS Identity and Access Management、Amazon CloudFront など) 経由でのみ使用できるサービスについては、設定済みリージョンを `us-east-1` に設定してクライアントをインスタンス化します。

> **Important**
>
> SDK には、マルチリージョンクライアントも付属しています。マルチリージョンクライアントでは、コマンドラインパラメーターとして指定されたパラメータ (`@region`) に基づいて、異なる AWS リージョンにリクエストをディスパッチできます。このクライアントでデフォルトで使用されるリージョンは、クライアントコンストラクタで指定する `region` オプションを使用して指定します。

クライアントのインスタンス化でコンストラクタを使用します

AWS SDK for PHP バージョン 3 では、クライアントをインスタンス化する方法が変更されています。バージョン 2 での `factory` メソッドの代わりに、`new` キーワードを使用するだけでクライアントをインスタンス化できます。

```
use Aws\DynamoDb\DynamoDbClient;

// Version 2 style
$client = DynamoDbClient::factory([
    'region'  => 'us-east-2'
]);

// Version 3 style
```

```
$client = new DynamoDbClient([
    'region'  => 'us-east-2',
    'version' => '2012-08-10'
]);
```

> **Note**
>
> `factory()` メソッドを使用したクライアントのインスタンス化も引き続き機能します。ただし、この方法は非推奨と見なされています。

クライアント設定が変更されています

AWS SDK for PHP バージョン 3 でのクライアント設定オプションは、バージョン 2 から少し変更されています。サポートされているすべてのオプションの説明については、「AWS SDK for PHP バージョン 3 の設定」ページを参照してください。

> **Important**
>
> バージョン 3 では、`'key'` と `'secret'` はルートレベルでは有効なオプションではなくなりましたが、`'credentials'` オプションの一部として渡すことはできます。この変更の 1 つの理由は、デベロッパーがプロジェクト内で AWS 認証情報をハードコーディングしないようにするためです。

Sdk オブジェクト

AWS SDK for PHP バージョン 3 では、`Aws\Sdk` オブジェクトが `Aws\Common\Aws` の置き換えとして導入されています。Sdk オブジェクトは、クライアントファクトリとして機能し、複数のクライアント間で共有される設定オプションを管理するために使用されます。

SDK バージョン 2 の `Aws` クラスはサービスロケーター (常に、クライアントの同じインスタンスを返す) のように機能していましたが、バージョン 3 のSdk クラスは、使用されるたびにクライアントの新しいインスタンスを返します。

また、Sdk オブジェクトでは、SDK バージョン 2 と同じ設定ファイル形式はサポートされていません。バージョン 2 の設定ファイル形式は、Guzzle 3 固有であったため、廃止されました。設定は、基本的な配列を使用してよりシンプルに行うことができ、詳細については「Sdk クラスの使用」に記載されています。

一部の API の結果が変更されています

SDK が API オペレーション、Amazon ElastiCache、Amazon RDS、および Amazon Redshift の結果を解析する方法に関して一貫性を保つために、一部の API レスポンスにラップ要素が追加されています。

例えば、Amazon RDS の DescribeEngineDefaultParameters 呼び出しの結果に、バージョン 3 では「EngineDefaults」ラップ要素が含まれています。この要素はバージョン 2 では存在しませんでした。

```
$client = new Aws\Rds\RdsClient([
    'region'  => 'us-west-1',
    'version' => '2014-09-01'
]);

// Version 2
$result = $client->describeEngineDefaultParameters();
$family = $result['DBParameterGroupFamily'];
$marker = $result['Marker'];

// Version 3
$result = $client->describeEngineDefaultParameters();
$family = $result['EngineDefaults']['DBParameterGroupFamily'];
$marker = $result['EngineDefaults']['Marker'];
```

以下のオペレーションが影響を受け、結果の出力にラップ要素が含まれるようになりました (括弧内は追加されたラップ要素)。

- Amazon ElastiCache
  - AuthorizeCacheSecurityGroupIngress (CacheSecurityGroup)
  - CopySnapshot (Snapshot)
  - CreateCacheCluster (CacheCluster)
  - CreateCacheParameterGroup (CacheParameterGroup)
  - CreateCacheSecurityGroup (CacheSecurityGroup)
  - CreateCacheSubnetGroup (CacheSubnetGroup)
  - CreateReplicationGroup (ReplicationGroup)
  - CreateSnapshot (Snapshot)
  - DeleteCacheCluster (CacheCluster)

- DeleteReplicationGroup (ReplicationGroup)
- DeleteSnapshot (Snapshot)
- DescribeEngineDefaultParameters (EngineDefaults)
- ModifyCacheCluster (CacheCluster)
- ModifyCacheSubnetGroup (CacheSubnetGroup)
- ModifyReplicationGroup (ReplicationGroup)
- PurchaseReservedCacheNodesOffering (ReservedCacheNode)
- RebootCacheCluster (CacheCluster)
- RevokeCacheSecurityGroupIngress (CacheSecurityGroup)

- Amazon RDS
  - AddSourceIdentifierToSubscription (EventSubscription)
  - AuthorizeDBSecurityGroupIngress (DBSecurityGroup)
  - CopyDBParameterGroup (DBParameterGroup)
  - CopyDBSnapshot (DBSnapshot)
  - CopyOptionGroup (OptionGroup)
  - CreateDBInstance (DBInstance)
  - CreateDBInstanceReadReplica (DBInstance)
  - CreateDBParameterGroup (DBParameterGroup)
  - CreateDBSecurityGroup (DBSecurityGroup)
  - CreateDBSnapshot (DBSnapshot)
  - CreateDBSubnetGroup (DBSubnetGroup)
  - CreateEventSubscription (EventSubscription)
  - CreateOptionGroup (OptionGroup)
  - DeleteDBInstance (DBInstance)
  - DeleteDBSnapshot (DBSnapshot)
  - DeleteEventSubscription (EventSubscription)
  - DescribeEngineDefaultParameters (EngineDefaults)
  - ModifyDBInstance (DBInstance)
  - ModifyDBSubnetGroup (DBSubnetGroup)
  - ModifyEventSubscription (EventSubscription)

- ModifyOptionGroup (OptionGroup)
- PromoteReadReplica (DBInstance)
- PurchaseReservedDBInstancesOffering (ReservedDBInstance)
- RebootDBInstance (DBInstance)
- RemoveSourceIdentifierFromSubscription (EventSubscription)
- RestoreDBInstanceFromDBSnapshot (DBInstance)
- RestoreDBInstanceToPointInTime (DBInstance)
- RevokeDBSecurityGroupIngress (DBSecurityGroup)

- Amazon Redshift
  - AuthorizeClusterSecurityGroupIngress (ClusterSecurityGroup)
  - AuthorizeSnapshotAccess (Snapshot)
  - CopyClusterSnapshot (Snapshot)
  - CreateCluster (Cluster)
  - CreateClusterParameterGroup (ClusterParameterGroup)
  - CreateClusterSecurityGroup (ClusterSecurityGroup)
  - CreateClusterSnapshot (Snapshot)
  - CreateClusterSubnetGroup (ClusterSubnetGroup)
  - CreateEventSubscription (EventSubscription)
  - CreateHsmClientCertificate (HsmClientCertificate)
  - CreateHsmConfiguration (HsmConfiguration)
  - DeleteCluster (Cluster)
  - DeleteClusterSnapshot (Snapshot)
  - DescribeDefaultClusterParameters (DefaultClusterParameters)
  - DisableSnapshotCopy (Cluster)
  - EnableSnapshotCopy (Cluster)
  - ModifyCluster (Cluster)
  - ModifyClusterSubnetGroup (ClusterSubnetGroup)
  - ModifyEventSubscription (EventSubscription)
  - ModifySnapshotCopyRetentionPeriod (Cluster)
  - PurchaseReservedNodeOffering (ReservedNode)

- RebootCluster (Cluster)
- RestoreFromClusterSnapshot (Cluster)
- RevokeClusterSecurityGroupIngress (ClusterSecurityGroup)
- RevokeSnapshotAccess (Snapshot)
- RotateEncryptionKey (Cluster)

Enum クラスが削除されました

AWS SDK for PHP バージョン 2 に存在していた Enum クラスを削除しました (例: `Aws\S3\Enum\CannedAcl`)。Enum は、有効なパラメーター値のグループを表す定数が含まれている、SDK のパブリック API 内の具象クラスでした。これらの列挙値は API バージョンに固有であり、時間の経過とともに変わる可能性があり、PHP の予約語と競合することがあり、あまり便利ではなくなったために、バージョン 3 では削除しました。これらは、バージョン 3 の特性に依存しないデータ駆動型と API バージョンをサポートしています。

Enum オブジェクトからの値を使用するのではなく、リテラル値を直接使用します (例: `CannedAcl::PUBLIC_READ` → `'public-read'`)。

きめ細かな例外クラスが削除されました

各サービスの名前空間に存在していた、きめ細かな例外クラス (例: `Aws\Rds\Exception\{SpecificError}Exception`) を、Enum を削除したのと同様の理由で削除しました。サービスまたはオペレーションによってスローされる例外は、使用されている API バージョンに依存します (バージョン間で変更されることがある)。また、特定のオペレーションでスローされる可能性がある例外の完全な一覧は提供していないため、バージョン 2 のきめ細かな例外クラスは不完全でした。

各サービスのルート例外クラス (例:`Aws\Rds\Exception\RdsException`) をキャッチしてエラーを処理します。例外の `getAwsErrorCode()` メソッドを使用して、特定のエラーコードをチェックできます。これは、別の例外クラスをキャッチするのと機能的には同じですが、SDK を肥大化せずにその関数を提供できます。

静的 Facade クラスが削除されました

AWS SDK for PHP バージョン 2 には、`Aws` クラスで `enableFacades()` を呼び出して各種サービスクライアントへの静的アクセスを有効にできる、Laravel から着想されたわかりにくい機能がありました。この機能は、PHP のベストプラクティスに反するものであり、ドキュメントに記載するのを 1 年以上前にやめました。バージョン 3 では、この機能は完全に削除されています。`Aws\Sdk` オ

ブジェクトからクライアントオブジェクトを取得して、それを静的クラスとしてではなくオブジェクトインスタンスとして使用します。

イテレーターはページネーターで置き換えられています

AWS SDK for PHP バージョン 2 には「イテレーター」という機能がありました。イテレーターは、ページ分割された結果を反復処理するために使用されるオブジェクトでした。イテレーターに関して、イテレーターでは各結果から特定の値のみが出力されるため、十分な柔軟性がないという苦情がありました。結果内の他の値が必要であっても、イベントリスナーを使用してその値を取得することはできませんでした。

バージョン 3 では、イテレーターはページネーターで置き換えられています。目的は似ていますが、ページネーターの方が柔軟性があります。それは、レスポンスから値ではなく、結果オブジェクトが生成されるためです。

以下の例は、バージョン 2 とバージョン 3 の両方で S3 ListObjects オペレーションのページ分割された結果を取得する方法を示すことで、ページネーターがイテレーターとどのように異なるかを示しています。

```
// Version 2
$objects = $s3Client->getIterator('ListObjects', ['Bucket' => 'my-bucket']);
foreach ($objects as $object) {
    echo $object['Key'] . "\n";
}
```

```
// Version 3
$results = $s3Client->getPaginator('ListObjects', ['Bucket' => 'my-bucket']);
foreach ($results as $result) {
    // You can extract any data that you want from the result.
    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
}
```

ページネーターオブジェクトには、search( )JMESPath 式を使用して、より簡単に結果セットからデータを抽出できる メソッドがあります。

```
$results = $s3Client->getPaginator('ListObjects', ['Bucket' => 'my-bucket']);
foreach ($results->search('Contents[].Key') as $key) {
    echo $key . "\n";
```

```
}
```

> **Note**
>
> バージョン 3 にスムーズに移行できるように、`getIterator()` メソッドは引き続きサポートされていますが、コードを移行してページネーターを使用することをお勧めします。

多くの高レベル抽象化が変更されています

全般的に、多くの高レベル抽象化 (クライアントは別として、サービス固有のヘルパーオブジェクト) が改善または更新されています。削除されたものもあります。

- Updated:
  - Amazon S3 マルチパートアップロードの使用方法が変更されました。Amazon S3 Glacier マルチパートアップロードも同様の方法で変更されています。
  - Amazon S3 署名済み URL の作成方法が変更されています。
  - `Aws\S3\Sync` 名前空間が `Aws\S3\Transfer` クラスに置き換えられています。`S3Client::uploadDirectory()` メソッドと `S3Client::downloadBucket()` メソッドは引き続き利用できますが、オプションが異なっています。「AWS SDK for PHP バージョン 3 での Amazon S3 Transfer Manager」のドキュメントを参照してください。
  - `Aws\S3\Model\ClearBucket` と `Aws\S3\Model\DeleteObjectsBatch` が、`Aws\S3\BatchDelete` と `S3Client::deleteMatchingObjects()` に置き換えられています。
  - 「AWS SDK for PHP バージョン 3 での DynamoDB セッションハンドラーの使用」で説明しているオプションと動作が多少変更されています。
  - `Aws\DynamoDb\Model\BatchRequest` 名前空間が `Aws\DynamoDb\WriteRequestBatch` に置き換えられています。「DynamoDB WriteRequestBatch」のドキュメントを参照してください。
  - `Aws\Ses\SesClient` は、`SendRawEmail` オペレーションを使用するときに、`RawMessage` の base64 エンコーディングを処理するようになりました。
- 削除済み:
  - Amazon DynamoDB の `Item`、`Attribute`、`ItemIterator` クラス - これらのクラスはバージョン 2.7.0 ですでに非推奨になっています。

- Amazon SNS メッセージバリデーター - これは、依存関係として SDK を必要としない個別の軽量プロジェクトになりました。ただし、SDK の phar と ZIP のディストリビューションにはこのプロジェクトが含まれています。AWS PHP 開発ブログに、このプロジェクトの入門ガイドがあります。
- Amazon S3 の `AcpBuilder` とその関連オブジェクトが削除されました。

SDK の両方のバージョンのサンプルコードの比較

AWS SDK for PHP バージョン 3 での使用方法がバージョン 2 と異なる例を、以下にいくつか示しています。

例: Amazon S3 ListObjects オペレーション

SDK バージョン 2 から

```
<?php

require '/path/to/vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$s3 = S3Client::factory([
    'profile' => 'my-credential-profile',
    'region'  => 'us-east-1'
]);

try {
    $result = $s3->listObjects([
        'Bucket' => 'my-bucket-name',
        'Key'    => 'my-object-key'
    ]);

    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

SDK バージョン 3 から

主な相違点:

- クライアントのインスタンス化に new ではなくfactory() を使用します。
- インストール時に 'version' と 'region' オプションが必須です。

```
<?php

require '/path/to/vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$s3 = new S3Client([
    'profile' => 'my-credential-profile',
    'region'  => 'us-east-1',
    'version' => '2006-03-01'
]);

try {
    $result = $s3->listObjects([
        'Bucket' => 'my-bucket-name',
        'Key'    => 'my-object-key'
    ]);

    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

例: グローバル構成によるクライアントのインスタンス化

SDK バージョン 2 から

```
<?php return array(
    'includes' => array('_aws'),
    'services' => array(
        'default_settings' => array(
            'params' => array(
```

```
                'profile' => 'my_profile',
                'region'  => 'us-east-1'
            )
        ),
        'dynamodb' => array(
            'extends' => 'dynamodb',
            'params' => array(
                'region'  => 'us-west-2'
            )
        ),
    )
);
```

```
<?php

require '/path/to/vendor/autoload.php';

use Aws\Common\Aws;

$aws = Aws::factory('path/to/my/config.php');

$sqs = $aws->get('sqs');
// Note: SQS client will be configured for us-east-1.

$dynamodb = $aws->get('dynamodb');
// Note: DynamoDB client will be configured for us-west-2.
```

SDK バージョン 3 から

主な相違点:

- Aws\Common\Aws ではなく Aws\Sdk クラスを使用します。
- 設定ファイルはありません。代わりに、設定の配列を使用します。
- インストール時に 'version' オプションが必須です。
- create<Service>() ではなく get('<service>') メソッドを使用します。

```
<?php

require '/path/to/vendor/autoload.php';
```

```
$sdk = new Aws\Sdk([
    'profile' => 'my_profile',
    'region' => 'us-east-1',
    'version' => 'latest',
    'DynamoDb' => [
        'region' => 'us-west-2',
    ],
]);

$sqs = $sdk->createSqs();
// Note: Amazon SQS client will be configured for us-east-1.

$dynamodb = $sdk->createDynamoDb();
// Note: DynamoDB client will be configured for us-west-2.
```

共有 config および credentials ファイル

共有 AWS config および credentials ファイルは、AWS SDK for PHP の認証と設定を指定する最も一般的な方法です。これらのファイルを使用して、ツールやアプリケーションが AWS SDK や AWS Command Line Interface 全体で使用できる設定を保存します。

共有 AWS config および credentials ファイルとファイルはプレーンテキストファイルで、デフォルトではコンピュータの「home」にある .aws フォルダにあるという名前のフォルダに格納されます。このファイルの場所を確認するには、AWS SDK とツールのリファレンスガイドの「共有 config および credentials ファイルの場所」を参照してください。

これらのファイルに保存できるすべての設定については、AWS SDK およびツールリファレンスガイドの「構成と認証設定のリファレンス」を参照してください。このリファレンスでは、環境変数などの代替ソースからの設定を適用する場合の優先順位についても説明します。

名前付きプロファイル

共有 config ファイルと credentials ファイル内の設定は、特定のプロファイルに関連付けられます。複数のプロファイルを使用して、さまざまな設定構成を作成してさまざまなシナリオに適用できます。プロファイルの 1 つが default プロファイルとして指定され、使用するプロファイルを明示的に指定しない場合は自動的に使用されます。

名前付きプロファイルの設定について詳しくは、AWS SDK およびツールリファレンスガイドの「共有 config と credentials ファイル」を参照してください。

profile オプションを使用して、クライアントをインスタンス化するときに使用する名前付きプロファイルを指定できます。

```
use Aws\DynamoDb\DynamoDbClient;

// Instantiate a client with the credentials from the my_profile_name profile
$client = new DynamoDbClient([
    'profile' => 'my_profile_name',
    'region'  => 'us-west-2',
    'version' => 'latest'
]);
```

AWS SDK for PHP での AWS サービスの操作

以下のセクションには、AWS SDK for PHP を使用して AWS サービスを操作する方法を示す例、チュートリアル、タスク、ガイドが含まれています。

トピック

- AWS SDK for PHP バージョン 3 の機能とオプションを使用する
- AWS SDK for PHP のガイダンス付きのコードサンプル

AWS SDK for PHP バージョン 3 の機能とオプションを使用する

AWS SDK for PHP バージョン 3 では、AWS サービス API と連携するための追加機能やオプションがサポートされています。このセクションでは、これらのオプションを対象としています。

トピック

- AWS SDK for PHP バージョン 3 での DynamoDB セッションハンドラーの使用
- Amazon S3 の機能、オプション

AWS SDK for PHP バージョン 3 での DynamoDB セッションハンドラーの使用

DynamoDB セッションハンドラーは PHP 用カスタムセッションハンドラーで、デベロッパーは Amazon DynamoDB をセッションストアとして使用できます。DynamoDB をセッションのストレージに使用すると、ローカルファイルシステムからセッションを共有の場所に移動することで、分散されたウェブアプリケーションのセッション処理で発生する問題を抑制します。DynamoDB は高速で、スケーラブルで、設定が簡単で、データのレプリケーションを自動的に処理します。

DynamoDB セッションハンドラーは、`session_set_save_handler()` 関数を使用して DynamoDB オペレーションを PHP のネイティブセッション関数にフックして、代替を許可します。これには、セッションのロックやガベージコレクションなどの機能のサポートが含まれています。これは、PHP のデフォルトのセッションハンドラーの一部です。

DynamoDB サービスに関する詳細については、Amazon DynamoDB ホームページを参照してください。

基本的な使用法

ステップ 1: ハンドラーを登録する

まず、インスタンス化し、セッションハンドラーを登録します。

```
use Aws\DynamoDb\SessionHandler;

$dynamoDb = new Aws\DynamoDb\DynamoDbClient([
    'region'=>'us-east-1'  // Since version 3.277.10 of the SDK,
]);                        // the 'version' parameter defaults to 'latest'.

$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name' => 'sessions'
]);

$sessionHandler->register();
```

ステップ 2。セッションを格納するテーブルを作成する

実際にセッションハンドラーを使用する前に、セッションを格納するテーブルを作成する必要があります。Amazon DynamoDB 向け AWSAWS コンソールまたは AWS SDK for PHP を使用して、前もってこれを実行できます。

このテーブルを作成するときは、プライマリキーの名前として「id」を使用します。また、セッションの自動ガベージコレクションの利点を活用するため、「expires」属性を使用して有効期限属性を設定することもお勧めします。

ステップ 3。通常どおりに PHP セッションを使用する

ハンドラーが登録され、テーブルが完成したら、$_SESSION superglobal を使用して、セッションに対して読み書きを行うことができます。これは PHP のデフォルトセッションハンドラーを使用する場合と同じです。DynamoDB セッションハンドラーは DynamoDB とのやり取りをカプセル化および抽象化します。これにより、PHP のネイティブセッション関数とインターフェイスを使用できるようになります。

```
// Start the session
session_start();

// Alter the session data
$_SESSION['user.name'] = 'jeremy';
```

```
$_SESSION['user.role'] = 'admin';

// Close the session (optional, but recommended)
session_write_close();
```

構成

次のオプションを使用してセッションハンドラーの動作を設定することができます。すべてのオプションは必要に応じて利用できますが、デフォルトの内容を十分理解してください。

table_name

セッションを保存する DynamoDB テーブルの名前。これのデフォルトは `'sessions'` です。

hash_key

DynamoDB セッションテーブル内のハッシュキーの名前。これのデフォルトは `'id'` です。

data_attribute

セッションデータが格納されている DynamoDB セッションテーブルの属性の名前。これのデフォルトは `'data'` です。

data_attribute_type

セッションデータが格納されている DynamoDB セッションテーブルの属性のタイプ。このデフォルトは `['string']` ですが、オプションで `'binary'` に設定することができます。

session_lifetime

ガーベージコレクションを開始するまでの非アクティブなセッションの継続期間。指定されない場合は、使用される実際の継続期間値は `ini_get('session.gc_maxlifetime')` です。

session_lifetime_attribute

セッションの有効期限が格納されている DynamoDB セッションテーブルの属性の名前。これのデフォルトは `'expires'` です。

consistent_read

セッションハンドラーが `GetItem` オペレーションに対して整合性のある読み込みを使用するかどうかを示します。デフォルトは `true` です。

locking

セッションのロックを使用するかどうかを指定します。デフォルトは `false` です。

batch_config

ガベージコレクション中にバッチ削除を使用するように設定します。これらのオプションは、DynamoDB WriteRequestBatch オブジェクトに直接渡されます。SessionHandler::garbageCollect() 経由でガベージコレクションを手動でトリガーします。

max_lock_wait_time

セッションハンドラーが、放棄するまでロックの取得を待機する最大時間 (秒単位)。デフォルトは 10 で、セッションのロックでのみ使用されます。

min_lock_retry_microtime

セッションハンドラーが、ロックの取得を試みる間に待機する最小時間 (マイクロ秒単位)。デフォルトは、10000 で、セッションのロックでのみ使用されます。

max_lock_retry_microtime

セッションハンドラーが、ロックの取得を試みる間に待機する最大時間 (マイクロ秒単位)。デフォルトは、50000 で、セッションのロックでのみ使用されます。

セッションハンドラーを設定するには、ハンドラーをインスタンス化するときに設定オプションを指定します。次のコードでは、すべての設定オプションを指定した例を示します。

```
$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name'                    => 'sessions',
    'hash_key'                      => 'id',
    'data_attribute'                => 'data',
    'data_attribute_type'           => 'string',
    'session_lifetime'              => 3600,
    'session_lifetime_attribute'    => 'expires',
    'consistent_read'               => true,
    'locking'                       => false,
    'batch_config'                  => [],
    'max_lock_wait_time'            => 10,
    'min_lock_retry_microtime'      => 5000,
    'max_lock_retry_microtime'      => 50000,
]);
```

料金

データストレージとデータ転送料金を除き、DynamoDB の使用に関連するコストは、使用するテーブルのプロビジョンドスループット性能に基づいて計算されます (詳細については、「Amazon DynamoDB 料金」を参照)。スループットは、書き込み容量と読み込み容量のユニットで測定されます。Amazon DynamoDB ホームページでは次のように記載されています。

読み込みキャパシティーユニットは、サイズが 4 KB である項目に対する、1 秒あたり 1 回の強力な整合性のある読み込み (または 1 秒あたり 2 回の結果整合性のある読み込み) を表します。書き込みキャパシティーユニットは、サイズが 1 KB である項目に対する、1 秒あたり 1 回の書き込みを表します。

最終的に、セッションテーブルに必要なスループットとコストは、予想されるトラフィックおよびセッションサイズとの相関があります。以下の表では、各セッション関数に対して、DynamoDB テーブルで実行される読み取りおよび書き込みオペレーションの量を説明します。

`session_start()` 経由の読み込み	• 1 回の読み込みオペレーション (`consistent_read` が `false` の場合は 0.5 のみ)。 • (条件付き) 有効期限が切れている場合にセッションを削除する 1 回の書き込みオペレーション。
`session_start()` 経由の読み込み (セッションロックの使用)。	• 最低 1 回の 書き込みオペレーション。 • (条件付き) セッションのロックの獲得を試行するたびの追加書き込みオペレーション。ロックの待機時間と再試行オプションの設定に基づく。 • (条件付き) 有効期限が切れている場合にセッションを削除する 1 回の書き込みオペレーション。
`session_write_close()` 経由で書き込み	• 1 回の書き込みオペレーション
`session_destroy()` 経由で削除	• 1 回の書き込みオペレーション
ガベージコレクション	• 期限切れのセッションに対してスキャンするテーブル内のデータ 4 KB ごとに 0.5 回の読み取りオペレーション。

- 削除するための期限切れの項目あたり 1 回の書き込みオペレーション。

セッションのロック

DynamoDB セッションハンドラーは、PHP のデフォルトのセッションハンドラー動作を模倣するペシミスティックセッションロックをサポートします。デフォルトでは、DynamoDB セッションハンドラーはこの機能をオフにします。特に Ajax リクエストまたは iflame を使用するときに、アプリケーションがセッションにアクセスして、パフォーマンスのボトルネックとコスト上昇につながる可能性があるからです。有効にする前に、セッションのロックがアプリケーションで必要とされるかどうかを十分に検討します。

セッションのロックを有効にするには、`'locking'` をインスタンス化するときに、`true` オプションを `SessionHandler` に設定してください。

```
$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name' => 'sessions',
    'locking'    => true,
]);
```

ガベージコレクション

「expires」属性を使用して、DynamoDB テーブルで TTL 属性を設定します。これにより、セッションが自動的にガベージコレクションの対象になり、自分でガベージコレクションを実行する必要がなくなります。

または、DynamoDB セッションハンドラーは、一連の `Scan` と `BatchWriteItem` オペレーションを使用することでセッションのガベージコレクションをサポートしています。`Scan` オペレーション処理の仕様上、すべての期限切れのセッションを検索し、それらを削除するには、ガベージコレクションプロセスに多大なプロビジョンドスループットが必要です。

このため、自動化されたガベージコレクションはサポートされていません。消費スループットのバーストが残りのアプリケーションは中断しない、オフピーク時間中に実行するようにガベージコレクションのスケジュールを設定することがベストプラクティスです。たとえば、夜間の cron ジョブでガベージコレクションを実行するスクリプトをトリガーすることもできます。このスクリプトは、次のように何らかの処理を実行する必要があります。

```
$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name'   => 'sessions',
```

```
    'batch_config' => [
        'batch_size' => 25,
        'before' => function ($command) {
            echo "About to delete a batch of expired sessions.\n";
        }
    ]
]);

$sessionHandler->garbageCollect();
```

'before' オプションを 'batch_config' オペレーションの内部で使用して、ガベージコレクションプロセスによって実行される BatchWriteItem オペレーションに遅延を導入します。これにより、ガベージコレクションが完了するまでにかかる時間は増加しますが、ガベージコレクション中にプロビジョンドスループット性能内、または近くに抑えられるように、DynamoDB セッションハンドラーによって行われるリクエストを分散できます。

```
$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name'   => 'sessions',
    'batch_config' => [
        'before' => function ($command) {
            $command['@http']['delay'] = 5000;
        }
    ]
]);

$sessionHandler->garbageCollect();
```

ベストプラクティス

1. アプリケーションサーバーと同じリージョンまたは地理的に最も近い AWS リージョンにセッションテーブルを作成します。これにより、アプリケーションと DynamoDB データベース間で、レイテンシーが最小になります。
2. 使用するセッションテーブルのプロビジョンドスループット性能を慎重に選択します。アプリケーションへの予想されるトラフィックとセッションの予想サイズを考慮します。または、テーブルに対して「オンデマンド」読み取り/書き込みキャパシティーを使用します。
3. AWS マネジメントコンソールまたは Amazon CloudWatch で消費スループットをモニタリングし、アプリケーションのニーズに対応できるようにスループット設定を調整します。
4. セッションのサイズを小さく抑えます (理想的には、1 KB 未満)。セッションが小さいと、パフォーマンスが向上し、必要なプロビジョンドスループット性能が減ります。

5. アプリケーションで必要な場合を除き、セッションのロックを使用しないでください。
6. PHP の組み込みのセッションガベージコレクショントリガーを使用する代わりに、cron ジョブ、または別のスケジューリングメカニズムを使用してガベージコレクションのスケジュールを設定して、オフピークの時間帯中に実行します。便利な `'batch_config'` オプションを使用します。

必要な IAM 許可

DynamoDB セッションハンドラーを使用するには、設定された認証情報に、前のステップで作成した DynamoDB テーブルを使用するためのアクセス許可が必要です。次の IAM ポリシーには必要最小限のアクセス許可が含まれています。このポリシーを使用するには、リソース値を以前に作成したテーブルの Amazon リソースネーム (ARN) で置き換えます。IAM ユーザーポリシーの作成と添付の詳細については、「IAM ユーザーガイド」の「IAM ポリシーを管理する」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:Scan",
        "dynamodb:BatchWriteItem"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:dynamodb:<region>:<account-id>:table/<table-name>"
    }
  ]
}
```

Amazon S3 の機能、オプション

このトピックでは、Amazon S3 AWS SDK for PHP と連携するためにバージョン 3 で提供されるその他の機能とオプションについて説明します。

トピック

- AWS SDK for PHP バージョン 3 での Amazon S3 マルチリージョンクライアント
- AWS SDK for PHP バージョン 3 での Amazon S3 ストリームラッパー

- AWS SDK for PHP バージョン 3 での Amazon S3 Transfer Manager
- AWS SDK for PHP バージョン 3 による Amazon S3 クライアント側の暗号化
- による Amazon S3 チェックサム

AWS SDK for PHP バージョン 3 での Amazon S3 マルチリージョンクライアント

AWS SDK for PHP バージョン 3 では、任意のサービスで使用できる汎用マルチリージョンクライアントが提供されています。これにより、ユーザーが任意のコマンドに `@region` 入力パラメーターを指定することで、コマンドの送信先となる AWS リージョンを指定できるようになります。また、SDK では、特定の Amazon S3 エラーにインテリジェントに応答し、それに応じてコマンドを再ルーティングする、Amazon S3 用のマルチリージョンクライアントも提供されています。これにより、ユーザーは同じクライアントを使用して複数のリージョンと通信できるようになります。この機能は、AWS SDK for PHP バージョン 3 を使用する Amazon S3 ストリームラッパーで、バケットが複数のリージョンに分散している場合に特に役立ちます。

基本的な使用法

Amazon S3 クライアントの基本的な使用パターンは、標準 S3 クライアントとマルチリージョンクライアントのどちらを使用していても同じです。コマンドレベルでの使用法の違いは、`@region` 入力パラメータを使用して AWS リージョンを指定できることだけです。

```
// Create a multi-region S3 client
$s3Client = (new \Aws\Sdk)->createMultiRegionS3(['version' => 'latest']);

// You can also use the client constructor
$s3Client = new \Aws\S3\S3MultiRegionClient([
    'version' => 'latest',
    // Any Region specified while creating the client will be used as the
    // default Region
    'region' => 'us-west-2',
]);

// Get the contents of a bucket
$objects = $s3Client->listObjects(['Bucket' => $bucketName]);

// If you would like to specify the Region to which to send a command, do so
// by providing an @region parameter
$objects = $s3Client->listObjects([
    'Bucket' => $bucketName,
    '@region' => 'eu-west-1',
```

```
]);
```

> **Important**
>
> マルチリージョン Amazon S3 クライアントを使用する場合は、永続的なリダイレクト例外は発生しません。標準 Amazon S3 クライアントでは、コマンドが間違ったリージョンに送信されると `Aws\S3\Exception\PermanentRedirectException` のインスタンスがスローされます。マルチリージョンクライアントでは、その場合に例外は発生せず、コマンドは正しいリージョンに再ディスパッチされます。

バケットリージョンキャッシュ

マルチリージョン Amazon S3 クライアントでは、特定のバケットが存在する AWS リージョンの内部キャッシュが保持されます。デフォルトでは、クライアントごとに独自のインメモリキャッシュがあります。クライアント間またはプロセス間でキャッシュを共有するには、`Aws\CacheInterface` のインスタンスを `bucket_region_cache` としてマルチリージョンクライアントに提供します。

```
use Aws\DoctrineCacheAdapter;
use Aws\Sdk;
use Doctrine\Common\Cache\ApcuCache;

$sdk = new Aws\Sdk([
    'version' => 'latest',
    'region' => 'us-west-2',
    'S3' => [
        'bucket_region_cache' => new DoctrineCacheAdapter(new ApcuCache),
    ],
]);
```

AWS SDK for PHP バージョン 3 での Amazon S3 ストリームラッパー

Amazon S3 ストリームラッパーにより、組み込み PHP 関数 `file_get_contents`、`fopen`、`copy`、`rename`、`unlink`、`mkdir`、`rmdir` などを使用して Amazon S3 に対してデータの保存および取得ができます。

使用するために Amazon S3 ストリームラッパーを登録する必要があります。

```
$client = new Aws\S3\S3Client([/** options **/]);
```

```
// Register the stream wrapper from an S3Client object
$client->registerStreamWrapper();
```

これにより、s3:// プロトコルを使用して Amazon S3 に保存されたバケットとオブジェクトにアクセスできます。Amazon S3 ストリームラッパーは、バケット名を含む文字列を受け入れます。この名前にフォワードスラッシュとオプションのオブジェクトキーまたはプレフィックスが続きます (s3://<bucket>[/<key-or-prefix>])。

> **Note**
>
> ストリームラッパーは少なくともユーザーの読み取りアクセス許可があるオブジェクトとバケットを操作するために設計されています。つまり、ユーザーが操作する必要がある任意のオブジェクトで ListBucket、任意のバケットで GetObject を実行するアクセス許可がユーザーに必要です。このアクセス許可がユーザーにないユースケースでは、Amazon S3 クライアントオペレーションを直接使用することをお勧めします。

データをダウンロードする

file_get_contents を使用して、オブジェクトのコンテンツを取得できます。ただし、この関数では、オブジェクトのコンテンツ全体をメモリにロードすることに注意してください。

```
// Download the body of the "key" object in the "bucket" bucket
$data = file_get_contents('s3://bucket/key');
```

大きいファイルを操作するとき、または Amazon S3 からデータをストリーミングする必要がある場合は、fopen() を使用します。

```
// Open a stream in read-only mode
if ($stream = fopen('s3://bucket/key', 'r')) {
    // While the stream is still open
    while (!feof($stream)) {
        // Read 1,024 bytes from the stream
        echo fread($stream, 1024);
    }
    // Be sure to close the stream resource when you're done with it
    fclose($stream);
}
```

> **Note**
>
> ファイルの書き込みエラーは fflush の呼び出しが行われた場合にのみ返されます。これらのエラーはフラッシュされていない fclose が呼び出されたときに返されません。fclose の戻り値は、ストリームを閉じる場合は true です。内部 fflush の応答に何らかエラーがあっても関係ありません。これらのエラーは file_put_contents 呼び出し時にも返されません。これは PHP 実装の仕様のためです。

シーク可能なストリームを開く

「r」モードで開いたストリームは、ストリームからデータを読み取ることのみ可能です。さらに、デフォルトではシーク可能ではありません。これは、真にストリーミング方式でデータを Amazon S3 からダウンロードできるようにするためです。ここで、事前読み取りバイトをメモリにバッファする必要はありません。ストリームをシーク可能にする必要がある場合は、seekable を関数のストリームコンテキストオプションに渡すことができます。

```
$context = stream_context_create([
    's3' => ['seekable' => true]
]);

if ($stream = fopen('s3://bucket/key', 'r', false, $context)) {
    // Read bytes from the stream
    fread($stream, 1024);
    // Seek back to the beginning of the stream
    fseek($stream, 0);
    // Read the same bytes that were previously read
    fread($stream, 1024);
    fclose($stream);
}
```

シーク可能なストリームをオープンすると、事前に読み込んだバイト数のシークが可能になります。リモートサーバーからまだ読み取っていないバイト数までスキップすることはできません。事前に読み取ったデータの記憶を許可するには、ストリームデコレータを使用して、データを PHP の一時ストリームにバッファします。キャッシュされたデータの量が 2 MB を超えた場合、一時ストリーム内のデータはメモリからディスクに転送されます。seekable ストリームコンテキスト設定を使用して、Amazon S3 から大きなファイルをダウンロードする際は、このことを念頭に置いてください。

データをアップロードする

`file_put_contents()` を使用して Amazon S3 にデータをアップロードすることができます。

```
file_put_contents('s3://bucket/key', 'Hello!');
```

`fopen()` で「w」、「x」、または「a」ストリームアクセスモードを使用したストリーミングデータによりさらに大きなファイルをアップロードできます。Amazon S3 ストリームラッパーは、同時読み取りおよび書き込みストリーム (「r+」、「w+」など) をサポートしていません。これは、HTTP プロトコルが同時読み取りおよび書き込みを許可しないためです。

```
$stream = fopen('s3://bucket/key', 'w');
fwrite($stream, 'Hello!');
fclose($stream);
```

> **Note**
>
> Amazon S3 ではリクエストのペイロードが送信される前に Content-Length ヘッダーを指定する必要があります。したがって、`PutObject` オペレーションでアップロード対象のデータは、ストリームがフラッシュまたはクローズされるまで、PHP の一時ストリームを使用して内部にバッファされます。

> **Note**
>
> ファイルの書き込みエラーは `fflush` の呼び出しが行われた場合にのみ返されます。これらのエラーはフラッシュされていない `fclose` が呼び出されたときに返されません。`fclose` の戻り値は、ストリームを閉じる場合は `true` です。内部 `fflush` の応答に何らかエラーがあっても関係ありません。これらのエラーは `file_put_contents` 呼び出し時にも返されません。これは PHP 実装の仕様のためです。

fopen モード

PHP の fopen() 関数では `$mode` オプションを指定する必要があります。モードオプションでは、データをストリームに対して読み取りや書き込みを行うかどうか、ストリームを開くときに、ファイルが存在しているかどうかを指定します。

Amazon S3 ストリームラッパーは、Amazon S3 オブジェクトをターゲットとするストリームで次のモードをサポートしています。

r	オブジェクトの存在を前提にしている読み取り専用ストリーム。
w	書き込みのみのストリーム。このオブジェクトが存在している場合、上書きされます。
a	書き込みのみのストリーム。オブジェクトがすでに存在する場合は、一時的なストリームにダウンロードされ、ストリームへの書き込みが事前にアップロードされたデータに追加されます。
x	書き込みのみのストリーム。エラーは、オプションがすでに存在する場合に発生します。

その他のオブジェクト関数

ストリームラッパーにより、さまざまな組み込み PHP 関数は Amazon S3 などのカスタムシステムで使用できます。ここに示すのは、Amazon S3 に格納されるオブジェクトでユーザーが実行することができる Amazon S3 ストリームラッパー関数の一部です。

unlink()	バケットから 1 つのオブジェクトを削除します。
	`// Delete an object from a bucket`
	`unlink('s3://bucket/key');`
	`DeleteObject` オペレーションに使用可能なオプションで渡すことができ、オブジェクトが削除される方法を変更できます (特定のオブジェクトバージョンの指定など)。
	`// Delete a specific version of an object from a bucket`

	`unlink('s3://bucket/key', stream_context_create([ 's3' => ['VersionId' => '123'] ]);`
filesize()	オブジェクトのサイズを取得します。 `// Get the Content-Length of an object $size = filesize('s3://bucket/key', );`
is_file()	URL がファイルかどうかを確認します。 `if (is_file('s3://bucket/key')) { echo 'It is a file!'; }`
file_exists()	オブジェクトが存在するかどうかを確認します。 `if (file_exists('s3://bucket/key')) { echo 'It exists!'; }`
filetype()	URL がファイルまたはバケット (dir) にマッピングされるかどうかを確認します。
file()	行の配列にあるオブジェクトの内容をロードします。GetObject オペレーションに使用可能なオプションを渡して、ファイルのダウンロード方法を変更できます。
filemtime()	オブジェクトが最後に変更された日付を取得します。

オブジェクトの名前を()	オブジェクトをコピーし、元のオブジェクトを削除することで、オブジェクトの名前を変更します。CopyObject および DeleteObject オペレーションに使用可能なオプションをストリームコンテキストパラメーターに渡すことができ、オブジェクトのコピーおよび削除方法を変更できます。

> **Note**
>
> 通常、copy は Amazon S3 ストリームラッパーとともに使用できますが、PHP での copy 関数の内部処理のために一部のエラーが適切に報告されないことがあります。代わりに AwsS3ObjectCopier のインスタンスを使用することをお勧めします。

バケットとフォルダの操作

mkdir() を使用してバケットを操作する

Amazon S3 バケットを作成して参照できます。使用するファイルシステムのディレクトリの変更とトラバーサルを PHP で許可する方法と同様です。

以下にバケットの作成の例を挙げます。

```
mkdir('s3://my-bucket');
```

> **Note**
>
> 2023 年 4 月に、Amazon S3 は、すべての新規作成されたバケットについて、S3 ブロックパブリックアクセスを自動的に有効にし、S3 アクセスコントロールリストを無効にするようになりました。この変更は、StreamWrapper の mkdir 関数が権限や ACL とどのように連携するかにも影響します。詳細については、この「AWS の新機能」記事を参照してください。

ストリームにコンテキストオプションを mkdir() メソッドに渡して、バケットの作成方法を変更できます。CreateBucket オペレーションで利用可能なパラメータを使用します。

```
// Create a bucket in the EU (Ireland) Region
mkdir('s3://my-bucket', 0500, true,
        stream_context_create([
            's3' => ['LocationConstraint' => 'eu-west-1']
        ]));
```

`rmdir()` 関数を使用してバケットを削除できます。

```
// Delete a bucket
rmdir('s3://my-bucket);
```

> **Note**
>
> バケットは、空である場合に限り、削除できます。

`mkdir()` を使用してフォルダを操作します。

バケットを作成したら、`mkdir()` を使用して、ファイルシステムのフォルダのようにフォルダとして機能するオブジェクトを作成できます。

次のコードスニペットは、「my-folder」という名前のフォルダオブジェクトを「my-bucket」という名前の既存のバケットに追加します。フォワードスラッシュ (/) 文字を使用して、フォルダオブジェクト名をバケット名およびその他のフォルダ名と区切ります。

```
mkdir('s3://my-bucket/my-folder')
```

2023 年 4 月以降の権限変更に関する前述の注意事項は、フォルダーオブジェクトを作成するときにも関係します。このブログ投稿には、必要に応じて権限を調整する方法についての情報が記載されています。

次のスニペットに示すように、`rmdir()` 関数を使用して空のフォルダオブジェクトを削除します。

```
rmdir('s3://my-bucket/my-folder')
```

バケットの内容を一覧表示する

opendir()、readdir()、rewinddir()、closedir() PHP 関数を Amazon S3 ストリームラッパーで使用して、バケットの内容を検討します。ListObjects オペレーションで使用できるパラメーター

を、opendir() 関数に対するカスタムストリームコンテキストオプションとして渡し、オブジェクトを一覧表示する方法を変更できます。

```
$dir = "s3://bucket/";

if (is_dir($dir) && ($dh = opendir($dir))) {
    while (($file = readdir($dh)) !== false) {
        echo "filename: {$file} : filetype: " . filetype($dir . $file) . "\n";
    }
    closedir($dh);
}
```

PHP の RecursiveDirectoryIterator を使用し、バケット内のプレフィックスと各オブジェクトを再帰的にリストすることができます。

```
$dir = 's3://bucket';
$iterator = new RecursiveIteratorIterator(new RecursiveDirectoryIterator($dir));

foreach ($iterator as $file) {
    echo $file->getType() . ': ' . $file . "\n";
}
```

少ない HTTP リクエストで再帰的に、バケットの内容をリストする別の方法は、Aws \recursive_dir_iterator($path, $context = null) 関数を使用します。

```
<?php
require 'vendor/autoload.php';

$iter = Aws\recursive_dir_iterator('s3://bucket/key');
foreach ($iter as $filename) {
    echo $filename . "\n";
}
```

ストリームコンテキストオプション

ストリームラッパーで使用されるクライアント、またはカスタムストリームコンテキストオプションを渡すことによって、バケットやキーに関する事前にロードされた情報をキャッシュするために使用できるキャッシュをカスタマイズできます。

ストリームラッパーはオペレーションごとに次のストリームコンテキストオプションをサポートしています。

client

コマンドを実行するために使用する Aws\AwsClientInterface オブジェクトです。

cache

以前に取得したファイル統計情報をキャッシュするために使用する Aws\CacheInterface のインスタンスです。デフォルトでは、ストリームラッパーはインメモリ LRU キャッシュを使用します。

AWS SDK for PHP バージョン 3 での Amazon S3 Transfer Manager

AWS SDK for PHP の Amazon S3 Transfer Manager は、ディレクトリ全体を Amazon S3 バケットにアップロードするため、およびバケット全体をローカルディレクトリにダウンロードするために使用されます。

ローカルディレクトリの Amazon S3 へのアップロード

Aws\S3\Transfer オブジェクトは転送を実行するために使用されます。以下の例では、ローカルディレクトリのファイルを再帰的に Amazon S3 バケットにアップロードする方法を示します。

```
// Create an S3 client.
$client = new \Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'version' => '2006-03-01',
]);

// Where the files will be sourced from.
$source = '/path/to/source/files';

// Where the files will be transferred to.
$dest = 's3://bucket';

// Create a transfer object.
$manager = new \Aws\S3\Transfer($client, $source, $dest);

// Perform the transfer synchronously.
$manager->transfer();
```

この例では、Amazon S3 クライアントを作成し、Transfer オブジェクトを作成して、同期転送を実行します。前の例を使用して、転送の実行に必要な、最小限のコードの量を示します。転送オブ

ジェクトは非同期で転送を実行でき、転送のカスタマイズに使用できるさまざまな設定オプションがあります。

s3:// URI でキープレフィックスを指定することで、ローカルファイルを Amazon S3 バケットの「サブフォルダ」にアップロードできます。次の例では、ローカルディスク上のファイルを bucket バケットにアップロードして、foo キープレフィックスにファイルを保存します。

```
$source = '/path/to/source/files';
$dest = 's3://bucket/foo';
$manager = new \Aws\S3\Transfer($client, $source, $dest);
$manager->transfer();
```

Amazon S3 バケットのダウンロード

$source 引数を Amazon S3 の URI (例: s3://bucket) および $dest 引数をローカルディレクトリへのパスとして指定することで、ディスクのローカルディレクトリに Amazon S3 バケットを再帰的にダウンロードできます。

```
// Where the files will be sourced from.
$source = 's3://bucket';

// Where the files will be transferred to.
$dest = '/path/to/destination/dir';

$manager = new \Aws\S3\Transfer($client, $source, $dest);
$manager->transfer();
```

> **Note**
>
> バケット内にオブジェクトをダウンロードするときに、SDK は必要なディレクトリを自動的に作成します。

「疑似フォルダ」に保存されたオブジェクトのみをダウンロードするには、Amazon S3 URI でバケットの後にキープレフィックスに含めることができます。次の例では、指定されたバケットのキープレフィックス「/foo」に保存されたファイルのみをダウンロードします。

```
$source = 's3://bucket/foo';
$dest = '/path/to/destination/dir';
$manager = new \Aws\S3\Transfer($client, $source, $dest);
```

```
$manager->transfer();
```

構成

Transfer オブジェクトのコンストラクタでは次の引数を指定できます。

$client

転送の実行に使用する Aws\ClientInterface オブジェクト。

$source (文字列 | **Iterator**)

転送されるソースデータ。これは、ディスク上のローカルパス (例 :/path/to/files) または Amazon S3 バケット (例 :s3://bucket) を指すことができます。s3:// URI には、共通プレフィックスにあるオブジェクトのみを転送するために使用できるキープレフィックスも含めることができます。

$source 引数が Amazon S3 の URI である場合は、$dest 引数がローカルディレクトリである必要があります (逆の場合も同様)。

文字列値の提供だけでなく、絶対ファイル名を生成する \Iterator オブジェクトを提供することもできます。イテレーターを指定した場合、 連想配列で オプションを指定するbase_dir必要が$optionsあります。

$dest

ファイルが転送される送信先。$source 引数がディスク上のローカルパスである場合、$dest は Amazon S3 バケット URI (例: s3://bucket) である必要があります。$source 引数が Amazon S3 バケット URI である場合、$dest 引数はディスク上のローカルパスである必要があります。

$options

転送オプション の連想配列。有効な転送オプションは、次の通りです。

add_content_md5 (ブール)

true に設定すると、アップロードの MD5 チェックサムを計算します。

base_dir (文字列)

$source がイテレーターの場合、ソースの基本ディレクトリ。$source オプションが配列ではない場合、このオプションは無視されます。

before (callable)

それぞれの転送の前に呼び出すコールバック。このコールバックは、function (Aws \Command $command) {...} のような関数の署名を持っている必要があります。指定されるコマンドは GetObject、PutObject、CreateMultipartUpload、UploadPart、または CompleteMultipartUpload コマンドです。

mup_threshold (int)

PutObject の代わりにマルチパートアップロードを使用するときのサイズ (バイト単位)。デフォルトで 16777216 (16 MB) に設定されます。

concurrency (int, default=5)

同時にアップロードするファイル数。理想的な同時実行値は、アップロードされるファイルの数と、各ファイルの平均サイズで変わります。一般的に、小さいファイルでは多数の同時実行により利点がありますが、大きいファイルではありません。

debug (ブール)

転送に関するデバッグ情報を出力するには true に設定します。STDOUT に書き込む代わりに、特定のストリームに書き込むには fopen( ) リソースに設定します。

非同期転送

Transfer オブジェクトは GuzzleHttp\Promise\PromisorInterface のインスタンスです。つまり、転送は非同期的に行われ、オブジェクトの promise メソッドを呼び出すことで、処理が開始されます。

```
$source = '/path/to/source/files';
$dest = 's3://bucket';
$manager = new \Aws\S3\Transfer($client, $source, $dest);

// Initiate the transfer and get a promise.
$promise = $manager->promise();

// Do something when the transfer is complete using the then() method.
$promise->then(function () {
    echo 'Done!';
});
```

いずれかのファイルを転送できない場合、promise は拒否されます。promise の otherwise メソッドを使用して失敗した転送を非同期で処理することができます。エラーが発生したとき、呼び出され

るコールバックを otherwise 関数は受け付けます。コールバックは拒否に対する $reason を受け入れます。これは、通常 Aws\Exception\AwsException のインスタンスです (ただし任意のタイプの値はコールバックに配信可能)。

```
$promise->otherwise(function ($reason) {
    echo 'Transfer failed: ';
    var_dump($reason);
});
```

Transfer オブジェクトは promise を返すので、これらの転送は他の非同期 promise と同時に発生する可能性があります。

Transfer Manager のコマンドのカスタマイズ

転送マネージャによって実行されるオペレーションに対して、コンストラクタに渡すコールバックを介してオペレーションでカスタムオプションを設定できます。

```
$uploader = new Transfer($s3Client, $source, $dest, [
    'before' => function (\Aws\Command $command) {
        // Commands can vary for multipart uploads, so check which command
        // is being processed.
        if (in_array($command->getName(), ['PutObject', 'CreateMultipartUpload'])) {
            // Set custom cache-control metadata.
            $command['CacheControl'] = 'max-age=3600';
            // Apply a canned ACL.
            $command['ACL'] = strpos($command['Key'], 'CONFIDENTIAL') ### false
                ? 'public-read'
                : 'private';
        }
    },
]);
```

AWS SDK for PHP バージョン 3 による Amazon S3 クライアント側の暗号化

クライアント側の暗号化を使用すると、ユーザーの環境内でデータは直接暗号化と復号が実行されます。つまり、Amazon S3 に転送する前にこのデータは暗号化されるので、暗号処理を外部サービスに依存しないで済みます。新規に実装する場合は、非推奨の S3EncryptionClient と S3EncryptionMultipartUploader ではなく、S3EncryptionClientV2 と S3EncryptionMultipartUploaderV2 の使用をお勧めします。非推奨バージョンを使用している古い実装では、移行を試みることをお勧めします。S3EncryptionClientV2 は、レガシーの S3EncryptionClient で暗号化されたデータの復号化をサポートしています。

AWS SDK for PHP はエンベロープ暗号化を実装し、暗号化と復号に OpenSSL を使用します。この実装はこのサポート機能に適合する他の SDK と相互運用可能です。また、SDK の promise ベースの非同期ワークフローと互換性があります。

移行ガイド

非推奨のクライアントから新しいクライアントへの移行を考えている場合は、こちらの移行ガイドを参照してください。

設定

クライアント側の暗号化の使用を開始するには、次が必要です。

- AWS KMS 暗号化キー
- S3 バケット

サンプルコードを実行する前に、AWS の認証情報を設定します。AWS SDK for PHP バージョン 3 の認証情報を参照してください。

暗号化

`S3EncryptionClientV2` で暗号化されたオブジェクトをアップロードするには、標準の `PutObject` パラメータに加えて、さらに 3 つのパラメータが必要です。

- `'@KmsEncryptionContext'` はキーと値のペアで、暗号化されたオブジェクトに追加のセキュリティレイヤーを提供するために使用されます。暗号化クライアントは同じキーを渡す必要があります。これは、get 呼び出しで自動的に行われます。追加のコンテキストが必要ない場合は、空の配列を渡すことができます。
- `@CipherOptions` は、使用する暗号やキーサイズなど、暗号化の追加設定です。
- `@MaterialsProvider` は、暗号キーと初期化ベクトルの生成、および暗号キーの暗号化を処理するプロバイダーです。

```
use Aws\S3\S3Client;
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\Kms\KmsClient;
use Aws\Crypto\KmsMaterialsProviderV2;

 // Let's construct our S3EncryptionClient using an S3Client
```

```
$encryptionClient = new S3EncryptionClientV2(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV2(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
    'Cipher' => 'gcm',
    'KeySize' => 256,
    // Additional configuration options
];

$result = $encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
```

> **Note**
>
> Amazon S3 および AWS KMS ベースサービスのエラーに加えて、`'@CipherOptions'` が正しく設定されていない場合、スローされた `InvalidArgumentException` オブジェクトを受け取ることがあります。

復号

オブジェクトのダウンロードと復号化には、標準の `GetObject` パラメータに加えて、4 つの追加パラメータがあり、そのうち 2 つは必須です。クライアントは基本的な暗号オプションを検出します。

- **`'@SecurityProfile'`**: 'V2' に設定すると、V2 互換形式で暗号化されたオブジェクトのみ

  復号化できます。また、このパラメータを 'V2_AND_LEGACY' に設定すると、V1 互換形式で暗号化されたオブジェクトを復号化することができます。移行をサポートするには、@SecurityProfile を 'V2_AND_LEGACY' に設定します。'V2' は、新しいアプリケーションの開発でのみ使用します。
- **`'@MaterialsProvider'`** は、暗号キーと初期化ベクトルの生成、

  および暗号キーの暗号化を処理するプロバイダーです。
- **`'@KmsAllowDecryptWithAnyCmk'`**: (オプション) このパラメータを true に設定すると、

  MaterialsProvider のコンストラクタに KMS キー ID を提供しなくても復号化できるようになります。デフォルト値は false です。
- **`'@CipherOptions'`** (オプション) は、使用する暗号やキーサイズなど、

  暗号化の追加設定です。

```
$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => true,
    '@SecurityProfile' => 'V2_AND_LEGACY',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

> **Note**
>
> Amazon S3 および AWS KMS ベースサービスのエラーに加えて、`'@CipherOptions'` が正しく設定されていない場合、スローされた `InvalidArgumentException` オブジェクトを受け取ることがあります。

暗号設定

`'Cipher'` (文字列)

暗号化中に暗号化クライアントが使用する暗号メソッド。現時点では、'gcm' のみがサポートされています。

> **Important**
>
> PHP が バージョン 7.1 に更新されると GCM 暗号向け OpenSSL を使用して 暗号化および復号するために必要な追加のパラメーターが含まれます。PHP バージョン 7.0 以前では、GCM サポート用の polyfill が提供されており、暗号化クライアント `S3EncryptionClientV2` および `S3EncryptionMultipartUploaderV2` で使用されています。ただし、大きな入力のパフォーマンスは、PHP 7.1 以降のネイティブ実装を使用するよりも、polyfill を使用した方がはるかに遅くなるため、それらを効果的に使用するには、古い PHP バージョン環境のアップグレードが必要になる場合があります。

`'KeySize'` (int)

暗号化のために生成するコンテンツ暗号化キーの長さ。デフォルトは 256 ビットです。有効な設定オプションは 256 および 128 です。

`'Aad'` (文字列)

暗号化されたペイロードに含める「追加認証データ」(オプション)。この情報は復号時に検証されます。`Aad` は「GCM」暗号化を使用する場合にのみ使用できます。

> **Important**
>
> 追加の認証データはすべての AWS SDK でサポートされているわけではないので、他のSDKではこのパラメータを使用して暗号化されたファイルを復号化できない可能性があります。

メタデータ戦略

`Aws\Crypto\MetadataStrategyInterface` を実装するクラスのインスタンスを提供するオプションもあります。このシンプルなインターフェイスは、エンベロープ暗号化マテリア

ルを含む Aws\Crypto\MetadataEnvelope の保存とロードを処理します。SDK は 2 つのクラスを実装します。Aws\S3\Crypto\HeadersMetadataStrategy と Aws\S3\Crypto\InstructionFileMetadataStrategy です。HeadersMetadataStrategy がデフォルトで使用されます。

```
$strategy = new InstructionFileMetadataStrategy(
    $s3Client
);

$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@MetadataStrategy' => $strategy,
    '@KmsEncryptionContext' => [],
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => false,
    '@MaterialsProvider' => $materialsProvider,
    '@SecurityProfile' => 'V2',
    '@MetadataStrategy' => $strategy,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

HeadersMetadataStrategy と InstructionFileMetadataStrategy に対するクラス名の定数は、::class を呼び出すことによっても取得できます。

```
$result = $encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@MetadataStrategy' => HeadersMetadataStrategy::class,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
```

> **Note**
>
> インストラクションファイルがアップロードされた後に障害が発生している場合は、自動的に削除されます。

マルチパートアップロード

クライアント側の暗号化を使用してマルチパートアップロードを実行することもできます。アップロードする前に、Aws\S3\Crypto\S3EncryptionMultipartUploaderV2 は暗号化のためにソースストリームを準備します。1 つ作成すると Aws\S3\MultipartUploader と Aws\S3\Crypto\S3EncryptionClientV2 を使用した場合と同様の結果になります。S3EncryptionMultipartUploaderV2 は '@MetadataStrategy' オプションを S3EncryptionClientV2 と同様に処理でき、'@CipherOptions' 設定ですべて利用できます。

```
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV2(
    new KmsClient([
        'region' => 'us-east-1',
        'version' => 'latest',
        'profile' => 'default',
    ]),
    $kmsKeyId
);

$bucket = 'the-bucket-name';
$key = 'the-upload-key';
$cipherOptions = [
    'Cipher' => 'gcm'
    'KeySize' => 256,
    // Additional configuration options
];

$multipartUploader = new S3EncryptionMultipartUploaderV2(
    new S3Client([
        'region' => 'us-east-1',
        'version' => 'latest',
        'profile' => 'default',
    ]),
    fopen('large-file-to-encrypt.txt', 'r'),
    [
        '@MaterialsProvider' => $materialsProvider,
```

```
        '@CipherOptions' => $cipherOptions,
        'bucket' => $bucket,
        'key' => $key,
    ]
);
$multipartUploader->upload();
```

> **Note**
>
> Amazon S3 および AWS KMS ベースサービスのエラーに加えて、`'@CipherOptions'` が正しく設定されていない場合、スローされた `InvalidArgumentException` オブジェクトを受け取ることがあります。

による Amazon S3 チェックサム

Amazon Simple Storage Service (Amazon S3) では、オブジェクトをアップロードするときにチェックサムを指定できます。チェックサムを指定すると、そのチェックサムはオブジェクトとともに保存され、オブジェクトのダウンロード時に検証できます。

チェックサムは、ファイルを転送する際のデータの整合性をさらに強化します。チェックサムを使用すると、受信したファイルが元のファイルと一致することを確認することで、データ整合性を検証できます。Amazon S3 のチェックサムに関する詳細は、「Amazon Simple Storage Service ユーザーガイド」を参照してください。

Amazon S3 は現在、SHA-1、SHA-256、CRC-32、CRC-32C の 4 つのチェックサムアルゴリズムをサポートしています。ニーズに最適なアルゴリズムを柔軟に選択して、SDK にチェックサムを計算させることができます。または、サポートされている 4 つのアルゴリズムのいずれかを使用して、事前に計算された独自のチェックサム値を指定することもできます。

チェックサムについては、オブジェクトのアップロードとオブジェクトのダウンロードという 2 つのリクエストフェーズで説明します。

オブジェクトのアップロード

このアルゴリズムの有効な値は`CRC32`、`CRC32C`、`SHA1`、および `SHA256` です。

次のコードスニペットは、CRC-32 チェックサムを含むオブジェクトをアップロードするリクエストを示しています。SDK はリクエストを送信すると、CRC-32 チェックサムを計算してオブジェクトをアップロードします。Amazon S3 はオブジェクトと共にチェックサムを保存します。

SDK が計算するチェックサムが、Amazon S3 がリクエストを受信したときに計算するチェックサムと一致しない場合、エラーが返されます。

事前に計算されたチェックサム値を使用してください。

リクエストで事前に計算されたチェックサム値を指定すると、SDK による自動計算が無効になり、代わりに提供された値が使用されます。

次の例は、事前に計算された SHA-256 チェックサムを含むリクエストを示しています。

Amazon S3 が、指定されたアルゴリズムのチェックサム値が正しくないと判断した場合、サービスはエラーレスポンスを返します。

マルチパートアップロード

チェックサムはマルチパートアップロードでも使用できます。

オブジェクトのダウンロード

getObject メソッドを使用してオブジェクトをダウンロードすると、SDK は

次のスニペット内のリクエストは、チェックサムを計算して値を比較することでレスポンス内のチェックサムを検証するよう SDK に指示します。

オブジェクトがチェックサム付きでアップロードされなかった場合、検証は行われません。

Amazon S3 のオブジェクトには複数のチェックサムを設定できますが、ダウンロード時に検証されるチェックサムは 1 つだけです。SDK が検証するチェックサムは、チェックサムアルゴリズムの効率性に基づいた次の優先順位によって決まります。

1. CRC-32C
2. CRC-32
3. SHA-1
4. SHA-256

たとえば、レスポンスに CRC-32 と SHA-256 の両方のチェックサムが含まれている場合、CRC-32 チェックサムのみが検証されます。

AWS SDK for PHP のガイダンス付きのコードサンプル

このセクションには、AWS を使用する一般的な AWS SDK for PHP シナリオを示すコードサンプルが含まれています。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

トピック

- AWS SDK for PHP バージョン 3 を使用した Amazon CloudWatch のコードサンプル
- AWS SDK for PHP バージョン 3 を使用したカスタム Amazon CloudSearch ドメインリクエストへの署名
- AWS SDK for PHP バージョン 3 を使用した Amazon CloudWatch の例
- AWS SDK for PHP バージョン 3 を使用した Amazon EC2 のコードサンプル
- AWS SDK for PHP バージョン 3 を使用して Amazon OpenSearch サービスの検索リクエストに署名する
- AWS Identity and Access Management バージョン 3 を使用した AWS SDK for PHP の例
- AWS Key Management Service バージョン 3 を使用した AWS SDK for PHP の例
- AWS SDK for PHP バージョン 3 を使用した Amazon Kinesis のコード例
- AWS Elemental MediaConvertAWS SDK for PHP バージョン 3 を使用した の例
- AWS SDK for PHP バージョン 3 を使用した Amazon S3 のコードサンプル
- Secrets Manager API および AWS SDK for PHP バージョン 3 を使用したシークレットの管理
- AWS SDK for PHP バージョン 3 を使用した Amazon SES のコードサンプル
- AWS SDK for PHP バージョン 3 を使用した Amazon SNS のコードサンプル
- AWS SDK for PHP バージョン 3 を使用した Amazon SQS のコード例
- Amazon EventBridge グローバルエンドポイントにイベントを送信する

AWS SDK for PHP バージョン 3 を使用した Amazon CloudWatch のコードサンプル

Amazon CloudFrontは、独自のウェブサーバーや Amazon S3 などの AWS サーバーから、静的および動的なウェブコンテンツの提供を高速化する AWS 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 で配信中のコンテンツをユーザーが要求すると、ユーザーは最もレイテンシーが少ないエッジロケーションにルーティングされます。コンテンツがすでにキャッシュされていない場合、CloudFront はオリジンサーバーからコピーを取得して配信し、今後のリクエストに備えてキャッシュします。

CloudFront の詳細については、Amazon CloudFront デベロッパーガイドを参照してください。

AWS SDK for PHP バージョン 3 用のすべてのサンプルコードは GitHub で入手できます。

CloudFront API と AWS SDK for PHP バージョン 3 を使用した Amazon CloudFront ディストリビューションの管理

Amazon は、世界中のエッジロケーションにコンテンツを CloudFront キャッシュして、独自のサーバー、または Amazon S3 や Amazon EC2 などの Amazon サービスに保存する静的ファイルと動的ファイルの配信を高速化します。ユーザーがウェブサイトからコンテンツをリクエストすると、ファイルがキャッシュされている場合、 は最も近いエッジロケーションからコンテンツを CloudFront 供給します。それ以外の場合は、ファイルのコピー CloudFront を取得して配信し、次のリクエストに備えてキャッシュします。エッジロケーションでコンテンツをキャッシュすることにより、そのエリアでの類似したユーザーリクエストのレイテンシーが減ります。

作成する CloudFront ディストリビューションごとに、コンテンツの場所と、ユーザーがリクエストを行ったときにコンテンツを配信する方法を指定します。このトピックでは、HTML、CSS、JSON、イメージファイルなどの静的ファイルおよび動的ファイルの配信について説明します。ビデオオンデマンド CloudFront で を使用する方法については、「 を使用したオンデマンドおよびライブストリーミングビデオ CloudFront」を参照してください。

以下の例では、次の方法を示しています。

- を使用してディストリビューションを作成しますCreateDistribution。
- を使用してディストリビューションを取得しますGetDistribution。
- を使用してディストリビューションを一覧表示しますListDistributions。
- を使用してディストリビューションを更新しますUpdateDistributions。

- を使用してディストリビューションを無効にしますDisableDistribution。
- を使用してディストリビューションを削除しますDeleteDistributions。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Amazon の使用の詳細については CloudFront、「Amazon CloudFront デベロッパーガイド」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

Amazon S3 バケットからディストリビューションを作成します。次の例では、オプションのパラメータはコメントアウトされていますが、デフォルト値は表示されています。ディストリビューションをカスタマイズするには、$distribution 内の値とパラメータの両方をコメント解除します。

CloudFront ディストリビューションを作成するには、 CreateDistributionオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
function createS3Distribution($cloudFrontClient, $distribution)
{
    try {
        $result = $cloudFrontClient->createDistribution([
            'DistributionConfig' => $distribution
        ]);

        $message = '';

        if (isset($result['Distribution']['Id'])) {
            $message = 'Distribution created with the ID of ' .
```

```
                $result['Distribution']['Id'];
        }

        $message .= ' and an effective URI of ' .
            $result['@metadata']['effectiveUri'] . '.';

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function createsTheS3Distribution()
{
    $originName = 'my-unique-origin-name';
    $s3BucketURL = 'my-bucket-name.s3.amazonaws.com';
    $callerReference = 'my-unique-caller-reference';
    $comment = 'my-comment-about-this-distribution';
    $defaultCacheBehavior = [
        'AllowedMethods' => [
            'CachedMethods' => [
                'Items' => ['HEAD', 'GET'],
                'Quantity' => 2
            ],
            'Items' => ['HEAD', 'GET'],
            'Quantity' => 2
        ],
        'Compress' => false,
        'DefaultTTL' => 0,
        'FieldLevelEncryptionId' => '',
        'ForwardedValues' => [
            'Cookies' => [
                'Forward' => 'none'
            ],
            'Headers' => [
                'Quantity' => 0
            ],
            'QueryString' => false,
            'QueryStringCacheKeys' => [
                'Quantity' => 0
            ]
        ],
        'LambdaFunctionAssociations' => ['Quantity' => 0],
        'MaxTTL' => 0,
```

```
        'MinTTL' => 0,
        'SmoothStreaming' => false,
        'TargetOriginId' => $originName,
        'TrustedSigners' => [
            'Enabled' => false,
            'Quantity' => 0
        ],
        'ViewerProtocolPolicy' => 'allow-all'
    ];
    $enabled = false;
    $origin = [
        'Items' => [
            [
                'DomainName' => $s3BucketURL,
                'Id' => $originName,
                'OriginPath' => '',
                'CustomHeaders' => ['Quantity' => 0],
                'S3OriginConfig' => ['OriginAccessIdentity' => '']
            ]
        ],
        'Quantity' => 1
    ];
    $distribution = [
        'CallerReference' => $callerReference,
        'Comment' => $comment,
        'DefaultCacheBehavior' => $defaultCacheBehavior,
        'Enabled' => $enabled,
        'Origins' => $origin
    ];

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo createS3Distribution($cloudFrontClient, $distribution);
}

// Uncomment the following line to run this code in an AWS account.
// createsTheS3Distribution();
```

CloudFront ディストリビューションを取得する

指定された CloudFront ディストリビューションのステータスと詳細を取得するには、GetDistributionオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
function getDistribution($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId
        ]);

        $message = '';

        if (isset($result['Distribution']['Status'])) {
            $message = 'The status of the distribution with the ID of ' .
                $result['Distribution']['Id'] . ' is currently ' .
                $result['Distribution']['Status'];
        }

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= ', and the effective URI is ' .
                $result['@metadata']['effectiveUri'] . '.';
        } else {
            $message = 'Error: Could not get the specified distribution. ' .
                'The distribution\'s status is not available.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}
```

```
function getsADistribution()
{
    $distributionId = 'E1BTGP2EXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo getDistribution($cloudFrontClient, $distributionId);
}

// Uncomment the following line to run this code in an AWS account.
// getsADistribution();
```

CloudFront ディストリビューションを一覧表示する

ListDistributions オペレーションを使用して、現在のアカウントから指定したAWSリージョンの既存の CloudFront ディストリビューションのリストを取得します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
function listDistributions($cloudFrontClient)
{
    try {
        $result = $cloudFrontClient->listDistributions([]);
        return $result;
    } catch (AwsException $e) {
        exit('Error: ' . $e->getAwsErrorMessage());
    }
}

function listTheDistributions()
{
    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
```

```
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-2'
    ]);

    $distributions = listDistributions($cloudFrontClient);

    if (count($distributions) == 0) {
        echo 'Could not find any distributions.';
    } else {
        foreach ($distributions['DistributionList']['Items'] as $distribution) {
            echo 'The distribution with the ID of ' . $distribution['Id'] .
                ' has the status of ' . $distribution['Status'] . '.' . "\n";
        }
    }
}

// Uncomment the following line to run this code in an AWS account.
// listTheDistributions();
```

CloudFront ディストリビューションを更新する

CloudFront ディストリビューションの更新は、ディストリビューションの作成と似ています。ただし、ディストリビューションを更新するときに、より多くのフィールドが必要になり、すべての値が含まれている必要があります。既存のディストリビューションを変更するには、最初に既存のディストリビューションを取得し、$distribution 配列内で変更する値を更新することをお勧めします。

指定された CloudFront ディストリビューションを更新するには、 UpdateDistributionオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function updateDistribution(
```

```
    $cloudFrontClient,
    $distributionId,
    $distributionConfig,
    $eTag
) {
    try {
        $result = $cloudFrontClient->updateDistribution([
            'DistributionConfig' => $distributionConfig,
            'Id' => $distributionId,
            'IfMatch' => $eTag
        ]);

        return 'The distribution with the following effective URI has ' .
            'been updated: ' . $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getDistributionConfig($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['Distribution']['DistributionConfig'])) {
            return [
                'DistributionConfig' => $result['Distribution']['DistributionConfig'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution configuration details.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}
```

```
function getDistributionETag($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['ETag'])) {
            return [
                'ETag' => $result['ETag'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution ETag header value.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function updateADistribution()
{
    // $distributionId = 'E1BTGP2EXAMPLE';
    $distributionId = 'E1X3BKQ569KEMH';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    // To change a distribution, you must first get the distribution's
    // ETag header value.
    $eTag = getDistributionETag($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $eTag)) {
        exit($eTag['Error']);
    }
```

```
    // To change a distribution, you must also first get information about
    // the distribution's current configuration. Then you must use that
    // information to build a new configuration.
    $currentConfig = getDistributionConfig($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $currentConfig)) {
        exit($currentConfig['Error']);
    }

    // To change a distribution's configuration, you can set the
    // distribution's related configuration value as part of a change request,
    // for example:
    // 'Enabled' => true
    // Some configuration values are required to be specified as part of a change
    // request, even if you don't plan to change their values. For ones you
    // don't want to change but are required to be specified, you can just reuse
    // their current values, as follows.
    $distributionConfig = [
        'CallerReference' => $currentConfig['DistributionConfig']["CallerReference"],
        'Comment' => $currentConfig['DistributionConfig']["Comment"],
        'DefaultCacheBehavior' => $currentConfig['DistributionConfig']
["DefaultCacheBehavior"],
        'DefaultRootObject' => $currentConfig['DistributionConfig']
["DefaultRootObject"],
        'Enabled' => $currentConfig['DistributionConfig']["Enabled"],
        'Origins' => $currentConfig['DistributionConfig']["Origins"],
        'Aliases' => $currentConfig['DistributionConfig']["Aliases"],
        'CustomErrorResponses' => $currentConfig['DistributionConfig']
["CustomErrorResponses"],
        'HttpVersion' => $currentConfig['DistributionConfig']["HttpVersion"],
        'CacheBehaviors' => $currentConfig['DistributionConfig']["CacheBehaviors"],
        'Logging' => $currentConfig['DistributionConfig']["Logging"],
        'PriceClass' => $currentConfig['DistributionConfig']["PriceClass"],
        'Restrictions' => $currentConfig['DistributionConfig']["Restrictions"],
        'ViewerCertificate' => $currentConfig['DistributionConfig']
["ViewerCertificate"],
        'WebACLId' => $currentConfig['DistributionConfig']["WebACLId"]
    ];

    echo updateDistribution(
        $cloudFrontClient,
        $distributionId,
        $distributionConfig,
        $eTag['ETag']
```

```
    );
}

// Uncomment the following line to run this code in an AWS account.
// updateADistribution();
```

CloudFront ディストリビューションを無効にする

ディストリビューションを無効化または削除するには、そのステータスをデプロイ済みから無効に変更します。

指定された CloudFront ディストリビューションを無効にするには、 DisableDistributionオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
function disableDistribution(
    $cloudFrontClient,
    $distributionId,
    $distributionConfig,
    $eTag
) {
    try {
        $result = $cloudFrontClient->updateDistribution([
            'DistributionConfig' => $distributionConfig,
            'Id' => $distributionId,
            'IfMatch' => $eTag
        ]);
        return 'The distribution with the following effective URI has ' .
            'been disabled: ' . $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getDistributionConfig($cloudFrontClient, $distributionId)
```

```
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['Distribution']['DistributionConfig'])) {
            return [
                'DistributionConfig' => $result['Distribution']['DistributionConfig'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution configuration details.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function getDistributionETag($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['ETag'])) {
            return [
                'ETag' => $result['ETag'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution ETag header value.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
```

```
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function disableADistribution()
{
    $distributionId = 'E1BTGP2EXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    // To disable a distribution, you must first get the distribution's
    // ETag header value.
    $eTag = getDistributionETag($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $eTag)) {
        exit($eTag['Error']);
    }

    // To delete a distribution, you must also first get information about
    // the distribution's current configuration. Then you must use that
    // information to build a new configuration, including setting the new
    // configuration to "disabled".
    $currentConfig = getDistributionConfig($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $currentConfig)) {
        exit($currentConfig['Error']);
    }

    $distributionConfig = [
        'CacheBehaviors' => $currentConfig['DistributionConfig']["CacheBehaviors"],
        'CallerReference' => $currentConfig['DistributionConfig']["CallerReference"],
        'Comment' => $currentConfig['DistributionConfig']["Comment"],
        'DefaultCacheBehavior' => $currentConfig['DistributionConfig']
["DefaultCacheBehavior"],
        'DefaultRootObject' => $currentConfig['DistributionConfig']
["DefaultRootObject"],
        'Enabled' => false,
        'Origins' => $currentConfig['DistributionConfig']["Origins"],
        'Aliases' => $currentConfig['DistributionConfig']["Aliases"],
```

```
        'CustomErrorResponses' => $currentConfig['DistributionConfig']
["CustomErrorResponses"],
        'HttpVersion' => $currentConfig['DistributionConfig']["HttpVersion"],
        'Logging' => $currentConfig['DistributionConfig']["Logging"],
        'PriceClass' => $currentConfig['DistributionConfig']["PriceClass"],
        'Restrictions' => $currentConfig['DistributionConfig']["Restrictions"],
        'ViewerCertificate' => $currentConfig['DistributionConfig']
["ViewerCertificate"],
        'WebACLId' => $currentConfig['DistributionConfig']["WebACLId"]
    ];

    echo disableDistribution(
        $cloudFrontClient,
        $distributionId,
        $distributionConfig,
        $eTag['ETag']
    );
}

// Uncomment the following line to run this code in an AWS account.
// disableADistribution();
```

CloudFront ディストリビューションを削除する

ディストリビューションのステータスが「無効」になったら、ディストリビューションを削除できます。

指定した CloudFront ディストリビューションを削除するには、 DeleteDistributionオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
function deleteDistribution($cloudFrontClient, $distributionId, $eTag)
{
    try {
        $result = $cloudFrontClient->deleteDistribution([
```

```
            'Id' => $distributionId,
            'IfMatch' => $eTag
        ]);
        return 'The distribution at the following effective URI has ' .
            'been deleted: ' . $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getDistributionETag($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['ETag'])) {
            return [
                'ETag' => $result['ETag'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution ETag header value.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function deleteADistribution()
{
    $distributionId = 'E17G7YNEXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);
```

```
    // To delete a distribution, you must first get the distribution's
    // ETag header value.
    $eTag = getDistributionETag($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $eTag)) {
        exit($eTag['Error']);
    } else {
        echo deleteDistribution(
            $cloudFrontClient,
            $distributionId,
            $eTag['ETag']
        );
    }
}

// Uncomment the following line to run this code in an AWS account.
// deleteADistribution();
```

CloudFront API と AWS SDK for PHP バージョン 3 を使用した Amazon CloudFront の無効化の管理

Amazon は、世界中のエッジロケーションに静的ファイルと動的ファイルのコピーを CloudFront キャッシュします。すべてのエッジロケーションでファイルを削除または更新するには、各ファイルまたはファイルのグループに対して無効化を作成します。

毎月最初の 1,000 件の無効化は無料です。 CloudFront エッジロケーションからコンテンツを削除する方法の詳細については、「ファイルの無効化」を参照してください。

以下の例では、次の方法を示しています。

- を使用してディストリビューションの無効化を作成しますCreateInvalidation。
- を使用してディストリビューションの無効化を取得しますGetInvalidation。
- を使用してディストリビューションを一覧表示しますListInvalidations。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Amazon の使用の詳細については CloudFront、「Amazon CloudFront デベロッパーガイド」を参照してください。

ディストリビューションの無効化の作成

削除する必要があるファイルのパスの場所を指定して、 CloudFront ディストリビューションの無効化を作成します。この例では、ディストリビューション内のすべてのファイルが無効になりますが、Items で特定のファイルを識別できます。

CloudFront ディストリビューションの無効化を作成するには、 CreateInvalidationオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
function createInvalidation(
    $cloudFrontClient,
    $distributionId,
    $callerReference,
    $paths,
    $quantity
) {
    try {
        $result = $cloudFrontClient->createInvalidation([
            'DistributionId' => $distributionId,
            'InvalidationBatch' => [
                'CallerReference' => $callerReference,
                'Paths' => [
                    'Items' => $paths,
                    'Quantity' => $quantity,
                ],
            ]
        ]);

        $message = '';
```

```
        if (isset($result['Location'])) {
            $message = 'The invalidation location is: ' . $result['Location'];
        }

        $message .= ' and the effective URI is ' . $result['@metadata']
['effectiveUri'] . '.';

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function createTheInvalidation()
{
    $distributionId = 'E17G7YNEXAMPLE';
    $callerReference = 'my-unique-value';
    $paths = ['/*'];
    $quantity = 1;

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo createInvalidation(
        $cloudFrontClient,
        $distributionId,
        $callerReference,
        $paths,
        $quantity
    );
}

// Uncomment the following line to run this code in an AWS account.
// createTheInvalidation();
```

ディストリビューションの無効化の取得

CloudFront ディストリビューションの無効化に関するステータスと詳細を取得するには、GetInvalidationオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
function getInvalidation($cloudFrontClient, $distributionId, $invalidationId)
{
    try {
        $result = $cloudFrontClient->getInvalidation([
            'DistributionId' => $distributionId,
            'Id' => $invalidationId,
        ]);

        $message = '';

        if (isset($result['Invalidation']['Status'])) {
            $message = 'The status for the invalidation with the ID of ' .
                $result['Invalidation']['Id'] . ' is ' .
                $result['Invalidation']['Status'];
        }

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= ', and the effective URI is ' .
                $result['@metadata']['effectiveUri'] . '.';
        } else {
            $message = 'Error: Could not get information about ' .
                'the invalidation. The invalidation\'s status ' .
                'was not available.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getsAnInvalidation()
{
    $distributionId = 'E1BTGP2EXAMPLE';
    $invalidationId = 'I1CDEZZEXAMPLE';
```

```
    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo getInvalidation($cloudFrontClient, $distributionId, $invalidationId);
}

// Uncomment the following line to run this code in an AWS account.
// getsAnInvalidation();
```

ディストリビューションの無効化のリスト表示

現在の CloudFront ディストリビューションの無効化をすべて一覧表示するには、 ListInvalidationsオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
function listInvalidations($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->listInvalidations([
            'DistributionId' => $distributionId
        ]);
        return $result;
    } catch (AwsException $e) {
        exit('Error: ' . $e->getAwsErrorMessage());
    }
}

function listTheInvalidations()
{
    $distributionId = 'E1WICG1EXAMPLE';
```

```
    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    $invalidations = listInvalidations(
        $cloudFrontClient,
        $distributionId
    );

    if (isset($invalidations['InvalidationList'])) {
        if ($invalidations['InvalidationList']['Quantity'] > 0) {
            foreach ($invalidations['InvalidationList']['Items'] as $invalidation) {
                echo 'The invalidation with the ID of ' . $invalidation['Id'] .
                    ' has the status of ' . $invalidation['Status'] . '.' . "\n";
            }
        } else {
            echo 'Could not find any invalidations for the specified distribution.';
        }
    } else {
        echo 'Error: Could not get invalidation information. Could not get ' .
            'information about the specified distribution.';
    }
}

// Uncomment the following line to run this code in an AWS account.
// listTheInvalidations();
```

AWS SDK for PHP バージョン 3 を使用した Amazon CloudFront URLs の署名

署名付き URL によりプライベートコンテンツへのアクセス権をユーザーに付与できます。署名付き URL には有効期限などの追加情報が含まれており、これによってコンテンツへのアクセスに対する管理が許可されます。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。プライベートディストリビューションを設定する方法と URL に署名 URLs」を参照してください。 CloudFront CloudFront

- getSignedURL を使用して署名付き Amazon CloudFront URL を作成します。 getSignedURL
- を使用して署名付き Amazon CloudFront Cookie を作成しますgetSignedCookie。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Amazon の使用の詳細については CloudFront、「Amazon CloudFront デベロッパーガイド」を参照してください。

プライベートディストリビューション CloudFront URLs の署名

SDK の CloudFront クライアントを使用して URL に署名できます。最初に、`CloudFrontClient` オブジェクトを作成する必要があります。既定ポリシーまたはカスタムポリシーを使用して、動画リソースの CloudFront URL に署名できます。

インポート

```
require 'vendor/autoload.php';

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function signPrivateDistribution(
    $cloudFrontClient,
    $resourceKey,
    $expires,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedUrl([
            'url' => $resourceKey,
            'expires' => $expires,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
```

```
}

function signAPrivateDistribution()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo signPrivateDistribution(
        $cloudFrontClient,
        $resourceKey,
        $expires,
        $privateKey,
        $keyPairId
    );
}

// Uncomment the following line to run this code in an AWS account.
// signAPrivateDistribution();
```

URL の作成 CloudFront URLs

カスタムポリシーを使用するには、policy キーを expires の代わりに指定します。

インポート

```
require 'vendor/autoload.php';

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function signPrivateDistributionPolicy(
    $cloudFrontClient,
```

```
    $resourceKey,
    $customPolicy,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedUrl([
            'url' => $resourceKey,
            'policy' => $customPolicy,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function signAPrivateDistributionPolicy()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $customPolicy = <<<POLICY
{
    "Statement": [
        {
            "Resource": "$resourceKey",
            "Condition": {
                "IpAddress": {"AWS:SourceIp": "{$_SERVER['REMOTE_ADDR']}/32"},
                "DateLessThan": {"AWS:EpochTime": $expires}
            }
        }
    ]
}
POLICY;
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);
```

```
    echo signPrivateDistributionPolicy(
        $cloudFrontClient,
        $resourceKey,
        $customPolicy,
        $privateKey,
        $keyPairId
    );
}

// Uncomment the following line to run this code in an AWS account.
// signAPrivateDistributionPolicy();
```

CloudFront 署名付き URL を使用する

署名付き URL の形式は、署名する URL で使用されているのが「HTTP」スキームであるか「RTMP」スキームであるかによって異なります。「HTTP」の場合は、完全な、絶対 URL が返されます。「RTMP」の場合は、わかりやすいように相対 URL だけが返されます。これは、一部のプレイヤーでホストとパスを個別パラメーターとして指定することが、必要となるからです。

次の例では、JWPlayer を使用してビデオを表示するウェブページを構築するために署名付き URL を使用する方法を示します。同じタイプの手法は、 などの他のプレイヤーにも適用されますがFlowPlayer、異なるクライアント側コードが必要です。

```
<html>
<head>
    <title>|CFlong| Streaming Example</title>
    <script type="text/javascript" src="https://example.com/jwplayer.js"></script>
</head>
<body>
    <div id="video">The canned policy video will be here.</div>
    <script type="text/javascript">
        jwplayer('video').setup({
            file: "<?= $streamHostUrl ?>/cfx/st/<?= $signedUrlCannedPolicy ?>",
            width: "720",
            height: "480"
        });
    </script>
</body>
</html>
```

プライベートディストリビューションの CloudFront Cookie の署名

署名付き URL の代わりに、署名付き Cookie を使用してプライベートディストリビューションにクライアントアクセスを許可することもできます。署名付き cookie により複数の制限されたファイル (HLS 形式の動画ファイルすべてやウェブサイトの購読者領域にあるすべてのファイルなど) へのアクセスを提供できます。署名付き URLs「Amazon CloudFront デベロッパーガイド」の「署名付き URLs」を参照してください。

署名付き Cookie の作成は、署名付き URL の作成に似ています。唯一の違いは、呼び出すメソッドです (`getSignedCookie` が `getSignedUrl` の代わりに呼び出される)。

インポート

```
require 'vendor/autoload.php';

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function signCookie(
    $cloudFrontClient,
    $resourceKey,
    $expires,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedCookie([
            'url' => $resourceKey,
            'expires' => $expires,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return [ 'Error' => $e->getAwsErrorMessage() ];
    }
}
```

```
function signACookie()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    $result = signCookie(
        $cloudFrontClient,
        $resourceKey,
        $expires,
        $privateKey,
        $keyPairId
    );

    /* If successful, returns something like:
    CloudFront-Expires = 1589926678
    CloudFront-Signature = Lv1DyC2q...2HPXaQ__
    CloudFront-Key-Pair-Id = AAPKAJIKZATYYYEXAMPLE
    */
    foreach ($result as $key => $value) {
        echo $key . ' = ' . $value . "\n";
    }
}

// Uncomment the following line to run this code in an AWS account.
// signACookie();
```

CloudFront Cookie の作成時にカスタムポリシーを使用する

`getSignedUrl` パラメーターの場合と同様に `'policy'` パラメーターを、`expires` パラメーターと `url` パラメーターの代わりに指定して、カスタムポリシーで Cookie に署名できます。カスタムポリシーを使用すると、リソースキーにワイルドカードを含めることができます。これにより、複数のファイルに対する単一の署名付き Cookie を作成できます。

`getSignedCookie` はキーと値のペアの配列を返します。それらのすべてを、プライベートディストリビューションへのアクセスを許可する Cookie として設定する必要があります。

インポート

```
require 'vendor/autoload.php';

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function signCookiePolicy(
    $cloudFrontClient,
    $customPolicy,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedCookie([
            'policy' => $customPolicy,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return [ 'Error' => $e->getAwsErrorMessage() ];
    }
}

function signACookiePolicy()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $customPolicy = <<<POLICY
{
    "Statement": [
        {
            "Resource": "{$resourceKey}",
            "Condition": {
                "IpAddress": {"AWS:SourceIp": "{$_SERVER['REMOTE_ADDR']}/32"},
                "DateLessThan": {"AWS:EpochTime": {$expires}}
            }
        }
```

```
    ]
}
POLICY;
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    $result = signCookiePolicy(
        $cloudFrontClient,
        $customPolicy,
        $privateKey,
        $keyPairId
    );

    /* If successful, returns something like:
    CloudFront-Policy = eyJTdGF0...fX19XX0_
    CloudFront-Signature = RowqEQWZ...N8vetw__
    CloudFront-Key-Pair-Id = AAPKAJIKZATYYYEXAMPLE
    */
    foreach ($result as $key => $value) {
        echo $key . ' = ' . $value . "\n";
    }
}

// Uncomment the following line to run this code in an AWS account.
// signACookiePolicy();
```

Guzzle クライアントに CloudFront Cookie を送信する

Guzzle クライアント使用するために、これらの Cookie を `GuzzleHttp\Cookie\CookieJar` に渡すこともできます。

```
use GuzzleHttp\Client;
use GuzzleHttp\Cookie\CookieJar;

$distribution = "example-distribution.cloudfront.net";
$client = new \GuzzleHttp\Client([
    'base_uri' => "https://$distribution",
```

```
    'cookies' => CookieJar::fromArray($signedCookieCustomPolicy, $distribution),
]);

$client->get('video.mp4');
```

詳細については、「Amazon CloudFront デベロッパーガイド」の「署名付き Cookie の使用」を参照してください。

AWS SDK for PHP バージョン 3 を使用したカスタム Amazon CloudSearch ドメインリクエストへの署名

Amazon CloudSearch ドメインリクエストは、 でサポートされている範囲を超えてカスタマイズできますAWS SDK for PHP。IAM 認証によって保護されているドメインへのカスタムリクエストを作成する必要がある場合は、SDK の認証情報プロバイダと署名者を使用して、任意の PSR-7 リクエストに署名できます。

たとえば、「Cloud Search の使用開始」に従って、ステップ 3 で IAM で保護されているドメインを使用する場合は、次のようにリクエストに署名して実行する必要があります。

以下の例では、次の方法を示しています。

- SignatureV4 を使用して、AWS 署名プロトコルでリクエストに署名します。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Amazon CloudSearch ドメインリクエストに署名する

インポート

```
require './vendor/autoload.php';

use Aws\Credentials\CredentialProvider;
use Aws\Signature\SignatureV4;
use GuzzleHttp\Client;
use GuzzleHttp\Psr7\Request;
```

サンプルコード

```
function searchDomain(
    $client,
    $domainName,
    $domainId,
    $domainRegion,
    $searchString
) {
    $domainPrefix = 'search-';
    $cloudSearchDomain = 'cloudsearch.amazonaws.com';
    $cloudSearchVersion = '2013-01-01';
    $searchPrefix = 'search?';

    // Specify the search to send.
    $request = new Request(
        'GET',
        "https://$domainPrefix$domainName-$domainId.$domainRegion." .
            "$cloudSearchDomain/$cloudSearchVersion/" .
            "$searchPrefix$searchString"
    );

    // Get default AWS account access credentials.
    $credentials = call_user_func(CredentialProvider::defaultProvider())->wait();

    // Sign the search request with the credentials.
    $signer = new SignatureV4('cloudsearch', $domainRegion);
    $request = $signer->signRequest($request, $credentials);

    // Send the signed search request.
    $response = $client->send($request);

    // Report the search results, if any.
    $results = json_decode($response->getBody());

    $message = '';

    if ($results->hits->found > 0) {
        $message .= 'Search results:' . "\n";

        foreach ($results->hits->hit as $hit) {
            $message .= $hit->fields->title . "\n";
        }
    } else {
```

```
        $message .= 'No search results.';
    }

    return $message;
}

function searchADomain()
{
    $domainName = 'my-search-domain';
    $domainId = '7kbitd6nyiglhdtmssxEXAMPLE';
    $domainRegion = 'us-east-1';
    $searchString = 'q=star+wars&return=title';
    $client = new Client();

    echo searchDomain(
        $client,
        $domainName,
        $domainId,
        $domainRegion,
        $searchString
    );
}

// Uncomment the following line to run this code in an AWS account.
// searchADomain();
```

AWS SDK for PHP バージョン 3 を使用した Amazon CloudWatch の例

Amazon CloudWatch （ CloudWatch) は、Amazon Web Services リソースと で実行しているアプリケーションを AWS リアルタイムでモニタリングするウェブサービスです。 CloudWatch を使用して、リソースとアプリケーションに対して測定できる変数であるメトリクスを収集および追跡できます。 CloudWatch アラームは、定義したルールに基づいて、モニタリングしているリソースに通知を送信するか、自動的に変更を加えます。

のすべてのサンプルコード AWS SDK for PHP は、 で GitHub入手できます。

認証情報

サンプルコードを実行する前に、「」の説明に従って AWS 認証情報を設定します認証情報。次に AWS SDK for PHP、「」の説明に従って をインポートします基本的な使用法。

トピック

- AWS SDK for PHP バージョン 3 での Amazon CloudWatch アラームの使用
- AWS SDK for PHP バージョン 3 CloudWatch を使用した Amazon からのメトリクスの取得
- AWS SDK for PHP バージョン 3 CloudWatch を使用した Amazon でのカスタムメトリクスの公開
- AWS SDK for PHP バージョン 3 を使用した Amazon CloudWatch Events へのイベントの送信
- AWS SDK for PHP バージョン 3 での Amazon アラームでの CloudWatch アラームアクションの使用

AWS SDK for PHP バージョン 3 での Amazon CloudWatch アラームの使用

Amazon CloudWatch アラームは、指定した期間にわたって 1 つのメトリクスを監視します。このアラームは、複数の期間にわたる一定のしきい値とメトリクスの値の関係性に基づき、1 つ以上のアクションを実行します。

以下の例では、次の方法を示しています。

- を使用してアラームを記述しますDescribeAlarms。
- を使用してアラームを作成しますPutMetricAlarm。
- を使用してアラームを削除しますDeleteAlarms。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

アラームの記述

インポート

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function describeAlarms($cloudWatchClient)
```

```
{
    try {
        $result = $cloudWatchClient->describeAlarms();

        $message = '';

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= 'Alarms at the effective URI of ' .
                $result['@metadata']['effectiveUri'] . "\n\n";

            if (isset($result['CompositeAlarms'])) {
                $message .= "Composite alarms:\n";

                foreach ($result['CompositeAlarms'] as $alarm) {
                    $message .= $alarm['AlarmName'] . "\n";
                }
            } else {
                $message .= "No composite alarms found.\n";
            }

            if (isset($result['MetricAlarms'])) {
                $message .= "Metric alarms:\n";

                foreach ($result['MetricAlarms'] as $alarm) {
                    $message .= $alarm['AlarmName'] . "\n";
                }
            } else {
                $message .= 'No metric alarms found.';
            }
        } else {
            $message .= 'No alarms found.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function describeTheAlarms()
{
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
```

```
        'version' => '2010-08-01'
    ]);

    echo describeAlarms($cloudWatchClient);
}

// Uncomment the following line to run this code in an AWS account.
// describeTheAlarms();
```

アラームの作成

インポート

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function putMetricAlarm(
    $cloudWatchClient,
    $cloudWatchRegion,
    $alarmName,
    $namespace,
    $metricName,
    $dimensions,
    $statistic,
    $period,
    $comparison,
    $threshold,
    $evaluationPeriods
) {
    try {
        $result = $cloudWatchClient->putMetricAlarm([
            'AlarmName' => $alarmName,
            'Namespace' => $namespace,
            'MetricName' => $metricName,
            'Dimensions' => $dimensions,
            'Statistic' => $statistic,
            'Period' => $period,
```

```
            'ComparisonOperator' => $comparison,
            'Threshold' => $threshold,
            'EvaluationPeriods' => $evaluationPeriods
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            if (
                $result['@metadata']['effectiveUri'] ==
                'https://monitoring.' . $cloudWatchRegion . '.amazonaws.com'
            ) {
                return 'Successfully created or updated specified alarm.';
            } else {
                return 'Could not create or update specified alarm.';
            }
        } else {
            return 'Could not create or update specified alarm.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function putTheMetricAlarm()
{
    $alarmName = 'my-ec2-resources';
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
        [
            'Name' => 'Resource',
            'Value' => 'vCPU'
        ],
        [
            'Name' => 'Service',
            'Value' => 'EC2'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
```

```
    ];
    $statistic = 'Average';
    $period = 300;
    $comparison = 'GreaterThanThreshold';
    $threshold = 1;
    $evaluationPeriods = 1;

    $cloudWatchRegion = 'us-east-1';
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => $cloudWatchRegion,
        'version' => '2010-08-01'
    ]);

    echo putMetricAlarm(
        $cloudWatchClient,
        $cloudWatchRegion,
        $alarmName,
        $namespace,
        $metricName,
        $dimensions,
        $statistic,
        $period,
        $comparison,
        $threshold,
        $evaluationPeriods
    );
}

// Uncomment the following line to run this code in an AWS account.
// putTheMetricAlarm();
```

アラームの削除

インポート

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function deleteAlarms($cloudWatchClient, $alarmNames)
{
    try {
        $result = $cloudWatchClient->deleteAlarms([
            'AlarmNames' => $alarmNames
        ]);

        return 'The specified alarms at the following effective URI have ' .
            'been deleted or do not currently exist: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function deleteTheAlarms()
{
    $alarmNames = array('my-alarm');

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo deleteAlarms($cloudWatchClient, $alarmNames);
}

// Uncomment the following line to run this code in an AWS account.
// deleteTheAlarms();
```

AWS SDK for PHP バージョン 3 CloudWatch を使用した Amazon からのメトリクスの取得

メトリクスとは、システムのパフォーマンスに関するデータです。Amazon EC2 インスタンスや、独自のアプリケーションメトリクスなどの一部のリソースの詳細モニタリングを有効にできます。

以下の例では、次の方法を示しています。

- を使用してメトリクスを一覧表示しますListMetrics。
- を使用してメトリクスのアラームを取得しますDescribeAlarmsForMetric。
- を使用して、指定されたメトリクスの統計を取得しますGetMetricStatistics。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

メトリクスの一覧表示

インポート

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function listMetrics($cloudWatchClient)
{
    try {
        $result = $cloudWatchClient->listMetrics();

        $message = '';

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= 'For the effective URI at ' .
                $result['@metadata']['effectiveUri'] . ":\n\n";

            if (
                (isset($result['Metrics'])) and
                (count($result['Metrics']) > 0)
            ) {
                $message .= "Metrics found:\n\n";

                foreach ($result['Metrics'] as $metric) {
                    $message .= 'For metric ' . $metric['MetricName'] .
                        ' in namespace ' . $metric['Namespace'] . ":\n";

                    if (
                        (isset($metric['Dimensions'])) and
```

```
                        (count($metric['Dimensions']) > 0)
                    ) {
                        $message .= "Dimensions:\n";

                        foreach ($metric['Dimensions'] as $dimension) {
                            $message .= 'Name: ' . $dimension['Name'] .
                                ', Value: ' . $dimension['Value'] . "\n";
                        }

                        $message .= "\n";
                    } else {
                        $message .= "No dimensions.\n\n";
                    }
                }
            } else {
                $message .= 'No metrics found.';
            }
        } else {
            $message .= 'No metrics found.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function listTheMetrics()
{
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo listMetrics($cloudWatchClient);
}

// Uncomment the following line to run this code in an AWS account.
// listTheMetrics();
```

メトリクスに対するアラームを取得する

インポート

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function describeAlarmsForMetric(
    $cloudWatchClient,
    $metricName,
    $namespace,
    $dimensions
) {
    try {
        $result = $cloudWatchClient->describeAlarmsForMetric([
            'MetricName' => $metricName,
            'Namespace' => $namespace,
            'Dimensions' => $dimensions
        ]);

        $message = '';

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] . ":\n\n";

            if (
                (isset($result['MetricAlarms'])) and
                (count($result['MetricAlarms']) > 0)
            ) {
                $message .= 'Matching alarms for ' . $metricName . ":\n\n";

                foreach ($result['MetricAlarms'] as $alarm) {
                    $message .= $alarm['AlarmName'] . "\n";
                }
            } else {
                $message .= 'No matching alarms found for ' . $metricName . '.';
            }
        } else {
            $message .= 'No matching alarms found for ' . $metricName . '.';
        }
```

```
        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function describeTheAlarmsForMetric()
{
    $metricName = 'BucketSizeBytes';
    $namespace = 'AWS/S3';
    $dimensions = [
        [
            'Name' => 'StorageType',
            'Value' => 'StandardStorage'
        ],
        [
            'Name' => 'BucketName',
            'Value' => 'my-bucket'
        ]
    ];

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo describeAlarmsForMetric(
        $cloudWatchClient,
        $metricName,
        $namespace,
        $dimensions
    );
}

// Uncomment the following line to run this code in an AWS account.
// describeTheAlarmsForMetric();
```

メトリクスの統計情報を取得する

インポート

```
require 'vendor/autoload.php';
```

```
use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function getMetricStatistics(
    $cloudWatchClient,
    $namespace,
    $metricName,
    $dimensions,
    $startTime,
    $endTime,
    $period,
    $statistics,
    $unit
) {
    try {
        $result = $cloudWatchClient->getMetricStatistics([
            'Namespace' => $namespace,
            'MetricName' => $metricName,
            'Dimensions' => $dimensions,
            'StartTime' => $startTime,
            'EndTime' => $endTime,
            'Period' => $period,
            'Statistics' => $statistics,
            'Unit' => $unit
        ]);

        $message = '';

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= 'For the effective URI at ' .
                $result['@metadata']['effectiveUri'] . "\n\n";

            if (
                (isset($result['Datapoints'])) and
                (count($result['Datapoints']) > 0)
            ) {
                $message .= "Datapoints found:\n\n";

                foreach ($result['Datapoints'] as $datapoint) {
                    foreach ($datapoint as $key => $value) {
```

```
                        $message .= $key . ' = ' . $value . "\n";
                    }

                    $message .= "\n";
                }
            } else {
                $message .= 'No datapoints found.';
            }
        } else {
            $message .= 'No datapoints found.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getTheMetricStatistics()
{
    // Average number of Amazon EC2 vCPUs every 5 minutes within
    // the past 3 hours.
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Service',
            'Value' => 'EC2'
        ],
        [
            'Name' => 'Resource',
            'Value' => 'vCPU'
        ],
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
    ];
    $startTime = strtotime('-3 hours');
    $endTime = strtotime('now');
```

```
$period = 300; // Seconds. (5 minutes = 300 seconds.)
$statistics = ['Average'];
$unit = 'None';

$cloudWatchClient = new CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-08-01'
]);

echo getMetricStatistics(
    $cloudWatchClient,
    $namespace,
    $metricName,
    $dimensions,
    $startTime,
    $endTime,
    $period,
    $statistics,
    $unit
);

// Another example: average number of bytes of standard storage in the
// specified Amazon S3 bucket each day for the past 3 days.

/*
$namespace = 'AWS/S3';
$metricName = 'BucketSizeBytes';
$dimensions = [
    [
        'Name' => 'StorageType',
        'Value'=> 'StandardStorage'
    ],
    [
        'Name' => 'BucketName',
        'Value' => 'my-bucket'
    ]
];
$startTime = strtotime('-3 days');
$endTime = strtotime('now');
$period = 86400; // Seconds. (1 day = 86400 seconds.)
$statistics = array('Average');
$unit = 'Bytes';
```

```
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo getMetricStatistics($cloudWatchClient, $namespace, $metricName,
    $dimensions, $startTime, $endTime, $period, $statistics, $unit);
    */
}

// Uncomment the following line to run this code in an AWS account.
// getTheMetricStatistics();
```

AWS SDK for PHP バージョン 3 CloudWatch を使用した Amazon でのカスタムメトリクスの公開

メトリクスとは、システムのパフォーマンスに関するデータです。1 つのアラームで、指定した期間中、1 つのメトリクスを監視します。このアラームは、複数の期間にわたる一定のしきい値とメトリクスの値の関係性に基づき、1 つ以上のアクションを実行します。

以下の例では、次の方法を示しています。

- を使用してメトリクスデータを公開しますPutMetricData。
- を使用してアラームを作成しますPutMetricAlarm。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

メトリクスデータを発行する

インポート

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function putMetricData(
    $cloudWatchClient,
    $cloudWatchRegion,
    $namespace,
    $metricData
) {
    try {
        $result = $cloudWatchClient->putMetricData([
            'Namespace' => $namespace,
            'MetricData' => $metricData
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            if (
                $result['@metadata']['effectiveUri'] ==
                'https://monitoring.' . $cloudWatchRegion . '.amazonaws.com'
            ) {
                return 'Successfully published datapoint(s).';
            } else {
                return 'Could not publish datapoint(s).';
            }
        } else {
            return 'Error: Could not publish datapoint(s).';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function putTheMetricData()
{
    $namespace = 'MyNamespace';
    $metricData = [
        [
            'MetricName' => 'MyMetric',
            'Timestamp' => 1589228818, // 11 May 2020, 20:26:58 UTC.
            'Dimensions' => [
                [
                    'Name' => 'MyDimension1',
```

```
                    'Value' => 'MyValue1'

                ],
                [
                    'Name' => 'MyDimension2',
                    'Value' => 'MyValue2'
                ]
            ],
            'Unit' => 'Count',
            'Value' => 1
        ]
    ];

    $cloudWatchRegion = 'us-east-1';
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => $cloudWatchRegion,
        'version' => '2010-08-01'
    ]);

    echo putMetricData(
        $cloudWatchClient,
        $cloudWatchRegion,
        $namespace,
        $metricData
    );
}

// Uncomment the following line to run this code in an AWS account.
// putTheMetricData();
```

アラームの作成

インポート

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function putMetricAlarm(
    $cloudWatchClient,
    $cloudWatchRegion,
    $alarmName,
    $namespace,
    $metricName,
    $dimensions,
    $statistic,
    $period,
    $comparison,
    $threshold,
    $evaluationPeriods
) {
    try {
        $result = $cloudWatchClient->putMetricAlarm([
            'AlarmName' => $alarmName,
            'Namespace' => $namespace,
            'MetricName' => $metricName,
            'Dimensions' => $dimensions,
            'Statistic' => $statistic,
            'Period' => $period,
            'ComparisonOperator' => $comparison,
            'Threshold' => $threshold,
            'EvaluationPeriods' => $evaluationPeriods
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            if (
                $result['@metadata']['effectiveUri'] ==
                'https://monitoring.' . $cloudWatchRegion . '.amazonaws.com'
            ) {
                return 'Successfully created or updated specified alarm.';
            } else {
                return 'Could not create or update specified alarm.';
            }
        } else {
            return 'Could not create or update specified alarm.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}
```

```
function putTheMetricAlarm()
{
    $alarmName = 'my-ec2-resources';
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
        [
            'Name' => 'Resource',
            'Value' => 'vCPU'
        ],
        [
            'Name' => 'Service',
            'Value' => 'EC2'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
    ];
    $statistic = 'Average';
    $period = 300;
    $comparison = 'GreaterThanThreshold';
    $threshold = 1;
    $evaluationPeriods = 1;

    $cloudWatchRegion = 'us-east-1';
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => $cloudWatchRegion,
        'version' => '2010-08-01'
    ]);

    echo putMetricAlarm(
        $cloudWatchClient,
        $cloudWatchRegion,
        $alarmName,
        $namespace,
        $metricName,
        $dimensions,
        $statistic,
```

```
        $period,
        $comparison,
        $threshold,
        $evaluationPeriods
    );
}

// Uncomment the following line to run this code in an AWS account.
// putTheMetricAlarm();
```

AWS SDK for PHP バージョン 3 を使用した Amazon CloudWatch Events へのイベントの送信

CloudWatch イベントは、Amazon Web Services (AWS) リソースの変更を示すシステムイベントのほぼリアルタイムのストリームをさまざまなターゲットに配信します。簡単なルールを使用して、一致したイベントを 1 つ以上のターゲット関数またはストリームに振り分けることができます。

以下の例では、次の方法を示しています。

- を使用してルールを作成しますPutRule。
- を使用してルールにターゲットを追加しますPutTargets。
- を使用してカスタムイベントを CloudWatch イベントに送信しますPutEvents。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

ルールの作成

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$client = new Aws\cloudwatchevents\cloudwatcheventsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2015-10-07'
]);

try {
    $result = $client->putRule([
        'Name' => 'DEMO_EVENT', // REQUIRED
        'RoleArn' => 'IAM_ROLE_ARN',
        'ScheduleExpression' => 'rate(5 minutes)',
        'State' => 'ENABLED',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

ルールにターゲットを追加する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$client = new Aws\cloudwatchevents\cloudwatcheventsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2015-10-07'
]);

try {
    $result = $client->putTargets([
        'Rule' => 'DEMO_EVENT', // REQUIRED
        'Targets' => [ // REQUIRED
```

```
            [
                'Arn' => 'LAMBDA_FUNCTION_ARN', // REQUIRED
                'Id' => 'myCloudWatchEventsTarget' // REQUIRED
            ],
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

カスタムイベントを送信する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$client = new Aws\cloudwatchevents\cloudwatcheventsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2015-10-07'
]);

try {
    $result = $client->putEvents([
        'Entries' => [ // REQUIRED
            [
                'Detail' => '<string>',
                'DetailType' => '<string>',
                'Resources' => ['<string>'],
                'Source' => '<string>',
                'Time' => time()
            ],
        ],
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 での Amazon アラームでの CloudWatch アラームアクションの使用

アラームアクションを使用して、Amazon EC2 インスタンスを自動的に停止、終了、再起動、または復旧するアラームを作成します。今後インスタンスを実行する必要がなくなったときに、停止または終了アクションを使用できます。再起動と復元アクションを使用して、自動的にそのインスタンスを再起動できます。

以下の例では、次の方法を示しています。

- を使用して、指定したアラームのアクションを有効にしますEnableAlarmActions。
- を使用して、指定したアラームのアクションを無効にしますDisableAlarmActions。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

アラームアクションの有効化

インポート

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function enableAlarmActions($cloudWatchClient, $alarmNames)
{
```

```
    try {
        $result = $cloudWatchClient->enableAlarmActions([
            'AlarmNames' => $alarmNames
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            return 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] .
                ', actions for any matching alarms have been enabled.';
        } else {
            return'Actions for some matching alarms ' .
                'might not have been enabled.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function enableTheAlarmActions()
{
    $alarmNames = array('my-alarm');

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo enableAlarmActions($cloudWatchClient, $alarmNames);
}

// Uncomment the following line to run this code in an AWS account.
// enableTheAlarmActions();
```

アラームアクションの無効化

インポート

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
function disableAlarmActions($cloudWatchClient, $alarmNames)
{
    try {
        $result = $cloudWatchClient->disableAlarmActions([
            'AlarmNames' => $alarmNames
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            return 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] .
                ', actions for any matching alarms have been disabled.';
        } else {
            return 'Actions for some matching alarms ' .
                'might not have been disabled.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function disableTheAlarmActions()
{
    $alarmNames = array('my-alarm');

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo disableAlarmActions($cloudWatchClient, $alarmNames);
}

// Uncomment the following line to run this code in an AWS account.
// disableTheAlarmActions();
```

AWS SDK for PHP バージョン 3 を使用した Amazon EC2 のコードサンプル

Amazon Elastic Compute Cloud (Amazon EC2) は、クラウド内で仮想サーバーのホスティングを提供するウェブサービスです。規模を変更可能なコンピューティング性能を提供することによって、ウェブスケールのクラウドコンピューティングをデベロッパーが簡単に利用できるように設計されています。

AWS SDK for PHP 用のすべてのサンプルコードは GitHub で入手できます。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

トピック

- AWS SDK for PHP バージョン 3 を使用した Amazon EC2 インスタンスの管理
- AWS SDK for PHP バージョン 3 を使用した Amazon EC2 での Elastic IP アドレスの使用
- AWS SDK for PHP バージョン 3 での Amazon EC2 のリージョンとアベイラビリティーゾーンの使用
- AWS SDK for PHP バージョン 3 を使用した Amazon EC2 キーペアでの作業
- AWS SDK for PHP バージョン 3 を使用した Amazon EC2 のセキュリティグループでの作業

AWS SDK for PHP バージョン 3 を使用した Amazon EC2 インスタンスの管理

以下の例では、次の方法を示しています。

- を使用して Amazon EC2 インスタンスを記述しますDescribeInstances。
- を使用して、実行中のインスタンスの詳細モニタリングを有効にしますMonitorInstances。
- を使用して、実行中のインスタンスのモニタリングを無効にしますUnmonitorInstances。
- を使用して、以前に停止した Amazon EBS-backed AMI を起動しますStartInstances。
- を使用して Amazon EBS-backed インスタンスを停止しますStopInstances。
- を使用して、1 つ以上のインスタンスの再起動をリクエストしますRebootInstances。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

インスタンスの説明

インポート

```
require 'vendor/autoload.php';

use Aws\Ec2\Ec2Client;
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);
$result = $ec2Client->describeInstances();
echo "Instances: \n";
foreach ($result['Reservations'] as $reservation) {
    foreach ($reservation['Instances'] as $instance) {
        echo "InstanceId: {$instance['InstanceId']} - {$instance['State']['Name']} \n";
    }
}
```

モニタリングを有効または無効にする

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
```

```
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$instanceIds = ['InstanceID1', 'InstanceID2'];

$monitorInstance = 'ON';

if ($monitorInstance == 'ON') {
    $result = $ec2Client->monitorInstances([
        'InstanceIds' => $instanceIds
    ]);
} else {
    $result = $ec2Client->unmonitorInstances([
        'InstanceIds' => $instanceIds
    ]);
}

var_dump($result);
```

インスタンスを起動または停止する

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$action = 'START';

$instanceIds = ['InstanceID1', 'InstanceID2'];

if ($action == 'START') {
```

```
    $result = $ec2Client->startInstances([
        'InstanceIds' => $instanceIds,
    ]);
} else {
    $result = $ec2Client->stopInstances([
        'InstanceIds' => $instanceIds,
    ]);
}

var_dump($result);
```

インスタンスの再起動

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$instanceIds = ['InstanceID1', 'InstanceID2'];

$result = $ec2Client->rebootInstances([
    'InstanceIds' => $instanceIds
]);

var_dump($result);
```

AWS SDK for PHP バージョン 3 を使用した Amazon EC2 での Elastic IP アドレスの使用

Elastic IP アドレスは、動的なクラウドコンピューティングのために設計された静的 IP アドレスです。Elastic IP アドレスは、AWS アカウント に関連付けられます。これは、インターネットから到

達可能なパブリック IP アドレスです。インスタンスにパブリック IP アドレスがない場合は、Elastic IP アドレスをインスタンスに関連付けて、インターネットとの通信を有効にできます。

以下の例では、次の方法を示しています。

- を使用して、1 つ以上のインスタンスを記述しますDescribeInstances。
- を使用して Elastic IP アドレスを取得しますAllocateAddress。
- を使用して、Elastic IP アドレスをインスタンスに関連付けますAssociateAddress。
- を使用して Elastic IP アドレスを解放しますReleaseAddress。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

インスタンスを記述する

インポート

```
require 'vendor/autoload.php';

use Aws\Ec2\Ec2Client;
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);
$result = $ec2Client->describeInstances();
echo "Instances: \n";
foreach ($result['Reservations'] as $reservation) {
    foreach ($reservation['Instances'] as $instance) {
        echo "InstanceId: {$instance['InstanceId']} - {$instance['State']['Name']} \n";
    }
```

```
}
```

アドレスを割り当てて関連付ける

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$instanceId = 'InstanceID';

$allocation = $ec2Client->allocateAddress(array(
    'DryRun' => false,
    'Domain' => 'vpc',
));

$result = $ec2Client->associateAddress(array(
    'DryRun' => false,
    'InstanceId' => $instanceId,
    'AllocationId' => $allocation->get('AllocationId')
));

var_dump($result);
```

アドレスを解放する

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$associationID = 'AssociationID';

$allocationID = 'AllocationID';

$result = $ec2Client->disassociateAddress([
    'AssociationId' => $associationID,
]);

$result = $ec2Client->releaseAddress([
    'AllocationId' => $allocationID,
]);

var_dump($result);
```

AWS SDK for PHP バージョン 3 での Amazon EC2 のリージョンとアベイラビリティーゾーンの使用

Amazon EC2 は、世界中の複数の口ケーションでホスティングされています。これらの場所は、AWS リージョンとアベイラビリティーゾーンから構成されています。各リージョンは地理的に離れた領域です。1 つのリージョンには複数の独立した口ケーションがあり、アベイラビリティーゾーンと呼ばれます。Amazon EC2 では、複数の口ケーションにインスタンスとデータを配置することができます。

以下の例では、次の方法を示しています。

- を使用して、使用可能なアベイラビリティーゾーンを記述しますDescribeAvailabilityZones。
- を使用して、現在利用可能なAWSリージョンを記述しますDescribeRegions。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

アベイラビリティーゾーンの詳細を表示する

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeAvailabilityZones();

var_dump($result);
```

リージョンの詳細を表示する

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);
```

```
$result = $ec2Client->describeRegions();

var_dump($result);
```

AWS SDK for PHP バージョン 3 を使用した Amazon EC2 キーペアでの作業

Amazon EC2 は公開キー暗号化を使用し、ログイン情報の暗号化と復号を行います。パブリックキー暗号は、パブリックキーを使用してデータを暗号化します。次に、受取人はプライベートキーを使用してデータを復号化します。パブリックキーとプライベートキーは、キーペアと呼ばれます。

以下の例では、次の方法を示しています。

- を使用して 2048 ビット RSA キーペアを作成しますCreateKeyPair。
- を使用して、指定されたキーペアを削除しますDeleteKeyPair。
- を使用して、1 つ以上のキーペアを記述しますDescribeKeyPairs。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

キーペアを作成する

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);
```

```
$keyPairName = 'my-keypair';

$result = $ec2Client->createKeyPair(array(
    'KeyName' => $keyPairName
));

// Save the private key
$saveKeyLocation = getenv('HOME') . "/.ssh/{$keyPairName}.pem";
file_put_contents($saveKeyLocation, $result['keyMaterial']);

// Update the key's permissions so it can be used with SSH
chmod($saveKeyLocation, 0600);
```

キーペアの削除

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$keyPairName = 'my-keypair';

$result = $ec2Client->deleteKeyPair(array(
    'KeyName' => $keyPairName
));

var_dump($result);
```

キーペアの記述

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeKeyPairs();

var_dump($result);
```

AWS SDK for PHP バージョン 3 を使用した Amazon EC2 のセキュリティグループでの作業

Amazon EC2 セキュリティグループは、1 つ以上のインスタンスのトラフィックを制御する仮想ファイアウォールとして機能します。各セキュリティグループに対してルールを追加し、関連付けられたインスタンスに対するトラフィックを許可します。セキュリティグループのルールはいつでも変更できます。新しいルールは、セキュリティグループに関連付けられたすべてのインスタンスに自動的に適用されます。

以下の例では、次の方法を示しています。

- を使用して、1 つ以上のセキュリティグループを記述しますDescribeSecurityGroups。
- を使用して、セキュリティグループに進入ルールを追加しますAuthorizeSecurityGroupIngress。
- を使用してセキュリティグループを作成しますCreateSecurityGroup。
- を使用してセキュリティグループを削除しますDeleteSecurityGroup。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

セキュリティグループを記述する

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeSecurityGroups();

var_dump($result);
```

進入ルールを追加する

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->authorizeSecurityGroupIngress(array(
    'GroupName' => 'string',
    'SourceSecurityGroupName' => 'string'
));
```

```
var_dump($result);
```

セキュリティグループの作成

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

// Create the security group
$securityGroupName = 'my-security-group';
$result = $ec2Client->createSecurityGroup(array(
    'GroupId' => $securityGroupName,

));

// Get the security group ID (optional)
$securityGroupId = $result->get('GroupId');

echo "Security Group ID: " . $securityGroupId . '\n';
```

セキュリティグループを削除する

インポート

```
require 'vendor/autoload.php';
```

サンプルコード

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$securityGroupId = 'my-security-group-id';

$result = $ec2Client->deleteSecurityGroup([
    'GroupId' => $securityGroupId
]);

var_dump($result);
```

AWS SDK for PHP バージョン 3 を使用して Amazon OpenSearch サービスの検索リクエストに署名する

Amazon OpenSearch Serviceは、一般的なオープンソースの検索および分析エンジンである Amazon OpenSearch Service のデプロイ、運用、およびスケーリングを容易にするマネージドサービスです。OpenSearch サービスは Amazon OpenSearch サービス API への直接アクセスを提供します。つまり、開発者は使い慣れたツール、および堅牢なセキュリティオプションを使用できます。Amazon OpenSearch Service のクライアントの多くでリクエスト署名がサポートされていますが、サポートされていないクライアントを使用している場合に、AWS SDK for PHP の組み込み認証情報プロバイダと署名者を使用して任意の PSR-7 リクエストに署名できます。

以下の例では、次の方法を示しています。

- SignatureV4 を使用して、AWS 署名プロトコルでリクエストに署名します。

AWS SDK for PHP 用のすべてのサンプルコードは GitHub で入手できます。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

OpenSearch Service リクエストの署名

OpenSearch Service は、Signature バージョン 4 を使用します。つまり、サービスの署名 (この場合は `es`) と OpenSearch Service ドメインの リージョンに対するリクエストに署名する必要があります。OpenSearch Service でサポートされているリージョンの一覧については、AWS の「Amazon Web Services 全般のリファレンスAWS のリージョンとエンドポイント」を参照してください。ただし、次の例では、`us-west-2` リージョンにある OpenSearch Service ドメインに対するリクエストに署名します。

必要な認証情報を指定するには、SDK のデフォルトプロバイダーチェーンを使用するか、「AWS SDK for PHP バージョン 3 の認証情報」に記載されている任意の形式の認証情報を使用することができます。また、PSR-7 リクエスト (次のコードでは、`$psr7Request` という名前を想定) も必要になります。

```
// Pull credentials from the default provider chain
$provider = Aws\Credentials\CredentialProvider::defaultProvider();
$credentials = call_user_func($provider)->wait();

// Create a signer with the service's signing name and Region
$signer = new Aws\Signature\SignatureV4('es', 'us-west-2');

// Sign your request
$signedRequest = $signer->signRequest($psr7Request, $credentials);
```

AWS Identity and Access Management バージョン 3 を使用した AWS SDK for PHP の例

AWS Identity and Access Management (IAM) は、Amazon Web Services のお客様が AWS でユーザーとユーザーの許可を管理できるようにするウェブサービスです。このサービスは、複数のユーザーまたはシステムがクラウドで AWS 製品を使用する組織を対象としています。IAM を使用すると、ユーザー、セキュリティ認証情報 (アクセスキーなど)、およびユーザーがアクセスできる AWS リソースを制御する許可を集中管理できます。

AWS SDK for PHP 用のすべてのサンプルコードは GitHub で入手できます。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

トピック

- AWS SDK for PHP バージョン 3 での IAM アクセスキーの管理
- AWS SDK for PHP バージョン 3 での IAM ユーザーの管理
- AWS SDK for PHP バージョン 3 での アカウントエイリアスの使用
- AWS SDK for PHP バージョン 3 での IAM ポリシーの使用
- AWS SDK for PHP バージョン 3 での サーバー証明書の使用

AWS SDK for PHP バージョン 3 での IAM アクセスキーの管理

ユーザーがプログラムで AWS を呼び出すには、独自のアクセスキーが必要です。このニーズを満たすために、IAM ユーザーのアクセスキー (アクセスキー ID およびシークレットアクセスキー) を作成、修正、表示、および更新できます。デフォルトでは、アクセスキーを作成したときのステータスが [Active] です。これは、ユーザーが API 呼び出しにそのアクセスキーを使用できることを意味します。

以下の例では、次の方法を示しています。

- を使用して、シークレットアクセスキーと対応するアクセスキー ID を作成しますCreateAccessKey。
- を使用して、IAM ユーザーに関連付けられたアクセスキー IDs に関する情報を返しますListAccessKeys。
- を使用して、アクセスキーが最後に使用された日時に関する情報を取得しますGetAccessKeyLastUsed。
- を使用して、アクセスキーのステータスをアクティブから非アクティブ、またはその逆に変更しますUpdateAccessKey。
- を使用して、IAM ユーザーに関連付けられたアクセスキーペアを削除しますDeleteAccessKey。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

アクセスキーの作成

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->createAccessKey([
        'UserName' => 'IAM_USER_NAME',
    ]);
    $keyID = $result['AccessKey']['AccessKeyId'];
    $createDate = $result['AccessKey']['CreateDate'];
    $userName = $result['AccessKey']['UserName'];
    $status = $result['AccessKey']['Status'];
    // $secretKey = $result['AccessKey']['SecretAccessKey']
    echo "<p>AccessKey " . $keyID . " created on " . $createDate . "</p>";
    echo "<p>Username: " . $userName . "</p>";
    echo "<p>Status: " . $status . "</p>";
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

アクセスキーの一覧表示

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->listAccessKeys();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

アクセスキーの前回使用時情報の取得

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->getAccessKeyLastUsed([
        'AccessKeyId' => 'ACCESS_KEY_ID', // REQUIRED
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

アクセスキーの更新

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->updateAccessKey([
        'AccessKeyId' => 'ACCESS_KEY_ID', // REQUIRED
        'Status' => 'Inactive', // REQUIRED
        'UserName' => 'IAM_USER_NAME',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

アクセスキーの削除

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteAccessKey([
        'AccessKeyId' => 'ACCESS_KEY_ID', // REQUIRED
        'UserName' => 'IAM_USER_NAME',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 での IAM ユーザーの管理

IAM ユーザーは AWS で作成するエンティティであり、AWS とやり取りするためにこれを使用する人またはサービスを表します。AWS のユーザーは名前と認証情報で構成されます。

以下の例では、次の方法を示しています。

- を使用して新しい IAM ユーザーを作成しますCreateUser。
- を使用して IAM ユーザーを一覧表示しますListUsers。
- を使用して IAM ユーザーを更新しますUpdateUser。
- を使用して IAM ユーザーに関する情報を取得しますGetUser。
- を使用して IAM ユーザーを削除しますDeleteUser。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

IAM ユーザーの作成

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->createUser(array(
        // UserName is required
        'UserName' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

IAM ユーザーのリストを取得する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->listUsers();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

IAM ユーザーを更新する

インポート

```

require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);
```

```
try {
    $result = $client->updateUser([
        // UserName is required
        'UserName' => 'string1',
        'NewUserName' => 'string'
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

IAM ユーザーに関する情報を取得する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->getUser([
        'UserName' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

IAM ユーザーを削除する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteUser([
        // UserName is required
        'UserName' => 'string'
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 での アカウントエイリアスの使用

サインインページの URL に、AWS アカウント ID ではなく企業の名前または他のわかりやすい識別子を含めるには、AWS アカウント ID のエイリアスを作成します。AWS アカウント エイリアスを作成すると、サインインページの URL は変更され、エイリアスが組み込まれます。

以下の例では、次の方法を示しています。

- を使用してエイリアスを作成しますCreateAccountAlias。
- AWS アカウント を使用して、 に関連付けられているエイリアスを一覧表示しますListAccountAliases。

- を使用してエイリアスを削除しますDeleteAccountAlias。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

エイリアスの作成

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->createAccountAlias(array(
        // AccountAlias is required
        'AccountAlias' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

アカウントエイリアスの一覧表示

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->listAccountAliases();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

エイリアスを削除する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
```

```
]);

try {
    $result = $client->deleteAccountAlias([
        // AccountAlias is required
        'AccountAlias' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 での IAM ポリシーの使用

ポリシーを作成することによって、ユーザーにアクセス許可を付与します。ポリシーとは、ユーザーが実行できるアクションと、そのアクションが影響を与えるリソースの一覧が記載されているドキュメントです。デフォルトでは、明示的に許可されていないアクションやリソースはすべて拒否されます。ポリシーを作成して、ユーザー、ユーザーのグループ、ユーザーが引き受けるロール、およびリソースにアタッチできます。

以下の例では、次の方法を示しています。

- を使用して管理ポリシーを作成しますCreatePolicy。
- を使用してポリシーをロールにアタッチしますAttachRolePolicy。
- を使用してポリシーをユーザーにアタッチしますAttachUserPolicy。
- を使用してポリシーをグループにアタッチしますAttachGroupPolicy。
- を使用してロールポリシーを削除しますDetachRolePolicy。
- を使用してユーザーポリシーを削除しますDetachUserPolicy。
- を使用してグループポリシーを削除しますDetachGroupPolicy。
- を使用して管理ポリシーを削除しますDeletePolicy。
- を使用してロールポリシーを削除しますDeleteRolePolicy。
- を使用してユーザーポリシーを削除しますDeleteUserPolicy。
- を使用してグループポリシーを削除しますDeleteGroupPolicy。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

ポリシーの作成

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

$myManagedPolicy = '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "RESOURCE_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
            "dynamodb:DeleteItem",
            "dynamodb:GetItem",
            "dynamodb:PutItem",
            "dynamodb:Scan",
            "dynamodb:UpdateItem"
        ],
            "Resource": "RESOURCE_ARN"
        }
    ]
```

```
}';

try {
    $result = $client->createPolicy(array(
        // PolicyName is required
        'PolicyName' => 'myDynamoDBPolicy',
        // PolicyDocument is required
        'PolicyDocument' => $myManagedPolicy
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

ポリシーをロールにアタッチする

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

$roleName = 'ROLE_NAME';

$policyName = 'AmazonDynamoDBFullAccess';

$policyArn = 'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess';

try {
    $attachedRolePolicies = $client->getIterator('ListAttachedRolePolicies', ([
```

```
        'RoleName' => $roleName,
    ]));
    if (count($attachedRolePolicies) > 0) {
        foreach ($attachedRolePolicies as $attachedRolePolicy) {
            if ($attachedRolePolicy['PolicyName'] == $policyName) {
                echo $policyName . " is already attached to this role. \n";
                exit();
            }
        }
    }
    $result = $client->attachRolePolicy(array(
        // RoleName is required
        'RoleName' => $roleName,
        // PolicyArn is required
        'PolicyArn' => $policyArn
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

ポリシーをユーザーにアタッチします

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

$userName = 'USER_NAME';
```

```
$policyName = 'AmazonDynamoDBFullAccess';

$policyArn = 'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess';

try {
    $attachedUserPolicies = $client->getIterator('ListAttachedUserPolicies', ([
        'UserName' => $userName,
    ]));
    if (count($attachedUserPolicies) > 0) {
        foreach ($attachedUserPolicies as $attachedUserPolicy) {
            if ($attachedUserPolicy['PolicyName'] == $policyName) {
                echo $policyName . " is already attached to this role. \n";
                exit();
            }
        }
    }
    $result = $client->attachUserPolicy(array(
        // UserName is required
        'UserName' => $userName,
        // PolicyArn is required
        'PolicyArn' => $policyArn,
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

グループにポリシーをアタッチする

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->attachGroupPolicy(array(
        // GroupName is required
        'GroupName' => 'string',
        // PolicyArn is required
        'PolicyArn' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

ユーザーポリシーをデタッチする

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->detachUserPolicy([
        // UserName is required
```

```
        'UserName' => 'string',
        // PolicyArn is required
        'PolicyArn' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

グループポリシーをデタッチする

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->detachGroupPolicy([
        // GroupName is required
        'GroupName' => 'string',
        // PolicyArn is required
        'PolicyArn' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

ポリシーの削除

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deletePolicy(array(
        // PolicyArn is required
        'PolicyArn' => 'string'
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

ロールポリシーを削除

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteRolePolicy([
        // RoleName is required
        'RoleName' => 'string',
        // PolicyName is required
        'PolicyName' => 'string'
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

ユーザーポリシーを削除する

インポート

```

require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteUserPolicy([
        // UserName is required
        'UserName' => 'string',
```

```
        // PolicyName is required
        'PolicyName' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

グループポリシーを削除する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteGroupPolicy(array(
        // GroupName is required
        'GroupName' => 'string',
        // PolicyName is required
        'PolicyName' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 での サーバー証明書の使用

AWS でウェブサイトまたはアプリケーションへの HTTPS 接続を有効にするには、SSL/TLS サーバー証明書が必要です。外部プロバイダーから取得した証明書を AWS でウェブサイトまたはアプリケーションで使用するには、証明書を IAM にアップロードするか、AWS Certificate Manager にインポートする必要があります。

以下の例では、次の方法を示しています。

- を使用して IAM に保存されている証明書を一覧表示しますListServerCertificates。
- を使用して証明書に関する情報を取得しますGetServerCertificate。
- を使用して証明書を更新しますUpdateServerCertificate。
- を使用して証明書を削除しますDeleteServerCertificate。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

サーバー証明書の一覧表示

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);
```

```
try {
    $result = $client->listServerCertificates();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

サーバー証明書を取得する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->getServerCertificate([
        // ServerCertificateName is required
        'ServerCertificateName' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

サーバー証明書の更新

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->updateServerCertificate([
        // ServerCertificateName is required
        'ServerCertificateName' => 'string',
        'NewServerCertificateName' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

サーバー証明書の削除

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

サンプルコード

```
$client = new IamClient([
```

```
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteServerCertificate([
        // ServerCertificateName is required
        'ServerCertificateName' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS Key Management Service バージョン 3 を使用した AWS SDK for PHP の例

AWS Key Management Service (AWS KMS) は、データの暗号化に使用される暗号化キーの作成と管理を容易にするマネージド型サービスです。AWS KMS に関する詳細は、Amazon KMS ドキュメントを参照してください。セキュア PHP のアプリケーションを作成しているか、他の AWS サービスにデータを送信しているかにかかわらず、AWS KMS を使用すると、キーを使用できるユーザー、および暗号化されたデータにアクセスできるユーザーを制御できます。

AWS SDK for PHP バージョン 3 用のすべてのサンプルコードは GitHub で入手できます。

トピック

- AWS KMS API および AWS SDK for PHP バージョン 3を使用したキーの操作
- AWS SDK for PHP バージョン 3 を使用した AWS KMS データキーの暗号化および復号化
- AWS SDK for PHP バージョン 3 を使用した AWS KMS キーポリシーの操作
- AWS KMS API および AWS SDK for PHP バージョン 3を使用した許可の操作
- AWS KMS API および AWS SDK for PHP バージョン 3を使用したエイリアスの操作

AWS KMS API および AWS SDK for PHP バージョン 3を使用したキーの操作

AWS Key Management Service (AWS KMS) の主なリソースは、AWS KMS keys です。KMS キーを使用してデータを暗号化できます。

以下の例では、次の方法を示しています。

- を使用してカスタマー KMS キーを作成しますCreateKey。
- を使用してデータキーを生成しますGenerateDataKey。
- を使用して KMS キーを表示しますDescribeKey。
- を使用して、KMS キーのキー IDs とキー ARN を取得しますListKeys。
- を使用して KMS キーを有効にしますEnableKey。
- を使用して KMS キーを無効にしますDisableKey。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

AWS Key Management Service (AWS KMS) の使用の詳細については、「AWS KMS デベロッパーガイド」を参照してください。

KMS キーを作成する

KMS キー を作成するには、 CreateKeyオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

//Creates a customer master key (CMK) in the caller's AWS account.
```

```
$desc = "Key for protecting critical data";

try {
    $result = $KmsClient->createKey([
        'Description' => $desc,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

データキーを生成する

データ暗号化キーを生成するには、 GenerateDataKeyオペレーションを使用します。このオペレーションでは、作成されるデータキーのプレーンテキストおよび暗号化されたコピーが返されます。データキーを生成する AWS KMS key を指定します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$keySpec = 'AES_256';

try {
    $result = $KmsClient->generateDataKey([
        'KeyId' => $keyId,
        'KeySpec' => $keySpec,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

KMS キーを表示する

KMS キーの Amazon リソースネーム (ARN) やキーステータス など、KMS キーに関する詳細情報を取得するには、 DescribeKeyオペレーションを使用します。

DescribeKey はエイリアスを取得しません。エイリアスを取得するには、 ListAliasesオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';

try {
    $result = $KmsClient->describeKey([
        'KeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
```

```
    echo "\n";
}
```

KMS キーのキー ID およびキー ARN を取得する

KMS キーの ID と ARN を取得するには、 ListAliasesオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$limit = 10;

try {
    $result = $KmsClient->listKeys([
        'Limit' => $limit,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

KMS キーを有効にする

無効な KMS キーを有効にするには、 EnableKeyオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';

try {
    $result = $KmsClient->enableKey([
        'KeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

KMS キーを無効にする

KMS キーを無効にするには、 DisableKeyオペレーションを使用します。KMS キーを無効にすると、使用されなくなります。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';

try {
    $result = $KmsClient->disableKey([
        'KeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS SDK for PHP バージョン 3 を使用した AWS KMS データキーの暗号化および復号化

データキーは、大量のデータや他のデータ暗号化キーといったデータを暗号化するための暗号化キーです。

AWS Key Management Service (AWS KMS) AWS KMS key を使用して、データキーを生成、暗号化、復号できます。

以下の例では、次の方法を示しています。

- Encrypt を使用してデータキーを暗号化する。
- Decrypt を使用してデータキーを復号化する。
- を使用して、新しい KMS キーでデータキーを再暗号化しますReEncrypt。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

AWS Key Management Service (AWS KMS) の使用の詳細については、「AWS KMS デベロッパーガイド」を参照してください。

暗号化

Encrypt オペレーションは、データキーを暗号化するように設計されていますが、頻繁には使用されていません。GenerateDataKey および GenerateDataKeyWithoutPlaintextオペレーションは、暗号化されたデータキーを返します。暗号化されたデータを新しい AWS リージョンに移動し、その新しいリージョンで KMS キーを使用してデータキーを暗号化するときに Encypt メソッドを使用できます。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$message = pack('c*', 1, 2, 3, 4, 5, 6, 7, 8, 9, 0);

try {
    $result = $KmsClient->encrypt([
        'KeyId' => $keyId,
        'Plaintext' => $message,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Decrypt

データキーを復号するには、Decrypt オペレーションを使用します。

`ciphertextBlob` 指定する は、GenerateDataKey、、GenerateDataKeyWithoutPlaintextまたは Encrypt レスポンスの `CiphertextBlob`フィールドの値である必要があります。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$ciphertext = 'Place your cipher text blob here';

try {
    $result = $KmsClient->decrypt([
        'CiphertextBlob' => $ciphertext,
    ]);
    $plaintext = $result['Plaintext'];
    var_dump($plaintext);
} catch (AwsException $e) {
    // Output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

再暗号化

暗号化されたデータキーを復号し、別の KMS キーでデータキーをすぐに再暗号化するには、ReEncryptオペレーションを使用します。このオペレーションは AWS KMS 内のサーバー側で完全に実行されるため、プレーンテキストが AWS KMS の外部に公開されることはありません。

ciphertextBlob 指定する は、GenerateDataKey、、GenerateDataKeyWithoutPlaintextまたは Encrypt レスポンスの CiphertextBlobフィールドの値である必要があります。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$ciphertextBlob = 'Place your cipher text blob here';

try {
    $result = $KmsClient->reEncrypt([
        'CiphertextBlob' => $ciphertextBlob,
        'DestinationKeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS SDK for PHP バージョン 3 を使用した AWS KMS キーポリシーの操作

AWS KMS key の作成時に、KMS キーを使用および管理できるユーザーを決定します。これらのアクセス許可は、キーポリシーと呼ばれるドキュメントに含まれます。キーポリシーを使って、カスタマー管理の KMS キーのアクセス権限をいつでも追加、削除、または変更できますが、AWS 管理の KMS キーのキーポリシーを編集することはできません。詳細については、「AWS KMS に対する認証とアクセスコントロール」を参照してください。

以下の例では、次の方法を示しています。

- を使用してキーポリシーの名前を一覧表示しますListKeyPolicies。
- を使用してキーポリシーを取得しますGetKeyPolicy。
- を使用してキーポリシーを設定しますPutKeyPolicy。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

AWS Key Management Service (AWS KMS) の使用の詳細については、「AWS KMS デベロッパーガイド」を参照してください。

すべてのキーポリシーをリストする

KMS キーのキーポリシーの名前を取得するには、`ListKeyPolicies` オペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$limit = 10;
```

```
try {
    $result = $KmsClient->listKeyPolicies([
        'KeyId' => $keyId,
        'Limit' => $limit,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

キーポリシーを取得する

KMS キーのキーポリシーを取得するには、GetKeyPolicy オペレーションを使用します。

GetKeyPolicy では、ポリシー名が必要です。KMS キーを作成したときにキーポリシーを作成した場合を除き、唯一の有効なポリシー名は default です。デフォルトキーポリシーの詳細については、AWS Key Management Service 開発者ガイドをご覧ください。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$policyName = "default";

try {
    $result = $KmsClient->getKeyPolicy([
```

```
        'KeyId' => $keyId,
        'PolicyName' => $policyName
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

キーポリシーを設定する

KMS キーのキーポリシーを作成または変更するには、PutKeyPolicy オペレーションを使用します。

PutKeyPolicy では、ポリシー名が必要です。KMS キーを作成したときにキーポリシーを作成した場合を除き、唯一の有効なポリシー名は default です。デフォルトキーポリシーの詳細については、AWS Key Management Service 開発者ガイドをご覧ください。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$policyName = "default";

try {
    $result = $KmsClient->putKeyPolicy([
        'KeyId' => $keyId,
```

```
        'PolicyName' => $policyName,
        'Policy' => '{
            "Version": "2012-10-17",
            "Id": "custom-policy-2016-12-07",
            "Statement": [
                { "Sid": "Enable IAM User Permissions",
                "Effect": "Allow",
                "Principal":
                   { "AWS": "arn:aws:iam::111122223333:user/root" },
                "Action": [ "kms:*" ],
                "Resource": "*" },
                { "Sid": "Enable IAM User Permissions",
                "Effect": "Allow",
                "Principal":
                   { "AWS": "arn:aws:iam::111122223333:user/ExampleUser" },
                "Action": [
                    "kms:Encrypt*",
                    "kms:GenerateDataKey*",
                    "kms:Decrypt*",
                    "kms:DescribeKey*",
                    "kms:ReEncrypt*"
                ],
                "Resource": "*" }
            ]
        } '
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS KMS API および AWS SDK for PHP バージョン 3を使用した許可の操作

許可は、アクセス許可を付与するための別のメカニズムです。キーポリシーに代わるものです。許可を使用すると、AWS プリンシパルで AWS Key Management Service (AWS KMS) カスタマー管理型の CMK を使用できる、長期のアクセスを付与することができます。AWS KMS keys詳細については、AWS Key Management Service デベロッパーガイドの「AWS KMS に付与する」を参照してください。

以下の例では、次の方法を示しています。

- を使用して KMS キーの許可を作成しますCreateGrant。
- を使用して KMS キーの許可を表示しますListGrants。
- を使用して KMS キーのグラントを廃止にしますRetireGrant。
- を使用して KMS キーの許可を取り消しますRevokeGrant。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

AWS Key Management Service (AWS KMS) の使用の詳細については、「AWS KMS デベロッパーガイド」を参照してください。

許可を作成する

の許可を作成するにはAWS KMS key、 CreateGrantオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$granteePrincipal = "arn:aws:iam::111122223333:user/Alice";
$operation = ['Encrypt', 'Decrypt']; // A list of operations that the grant allows.

try {
```

```
    $result = $KmsClient->createGrant([
        'GranteePrincipal' => $granteePrincipal,
        'KeyId' => $keyId,
        'Operations' => $operation
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

許可を表示する

の許可に関する詳細情報を取得するにはAWS KMS key、 ListGrantsオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$limit = 10;

try {
    $result = $KmsClient->listGrants([
        'KeyId' => $keyId,
        'Limit' => $limit,
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

許可を無効にする

の許可を使用停止にするにはAWS KMS key、 RetireGrantオペレーションを使用します。許可の使用が完了した後でクリーンアップを実行する場合に、許可を無効にします。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$grantToken = 'Place your grant token here';

try {
    $result = $KmsClient->retireGrant([
        'GrantToken' => $grantToken,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

//Can also identify grant to retire by a combination of the grant ID
//and the Amazon Resource Name (ARN) of the customer master key (CMK)
$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
```

```
$grantId = 'Unique identifier of the grant returned during CreateGrant operation';

try {
    $result = $KmsClient->retireGrant([
        'GrantId' => $grantToken,
        'KeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

許可を取り消す

に対する許可を取り消すにはAWS KMS key、 RevokeGrantオペレーションを使用します。許可を取り消して、許可に依存しているオペレーションを明示的に拒否することができます。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$grantId = "grant1";

try {
    $result = $KmsClient->revokeGrant([
        'KeyId' => $keyId,
        'GrantId' => $grantId,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS KMS API および AWS SDK for PHP バージョン 3を使用したエイリアスの操作

AWS Key Management Service(AWS KMS) は、エイリアスと呼ばれる AWS KMS key の表示名をオプションで提供します。

以下の例では、次の方法を示しています。

- を使用してエイリアスを作成しますCreateAlias。
- を使用してエイリアスを表示しますListAliases。
- を使用してエイリアスを更新しますUpdateAlias。
- を使用してエイリアスを削除しますDeleteAlias。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

AWS Key Management Service (AWS KMS) の使用の詳細については、「AWS KMS デベロッパーガイド」を参照してください。

エイリアスの作成

KMS キーのエイリアスを作成するには、 CreateAliasオペレーションを使用します。エイリアスはアカウントと AWS リージョンで一意であることが必要です。既にエイリアスがある KMS キー用にエイリアスを作成した場合、`CreateAlias` によって同じ KMS キーに対して別のエイリアスが作成されます。既存のエイリアスは置き換えられません。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$aliasName = "alias/projectKey1";

try {
    $result = $KmsClient->createAlias([
        'AliasName' => $aliasName,
        'TargetKeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

エイリアスを表示する

呼び出し元の AWS アカウントおよび のすべてのエイリアスを一覧表示するにはAWS リージョン、ListAliasesオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$limit = 10;

try {
    $result = $KmsClient->listAliases([
        'Limit' => $limit,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

エイリアスを更新する

既存のエイリアスを別の KMS キーに関連付けるには、 UpdateAliasオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);
```

```
$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$aliasName = "alias/projectKey1";

try {
    $result = $KmsClient->updateAlias([
        'AliasName' => $aliasName,
        'TargetKeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

エイリアスを削除する

エイリアスを削除するには、 DeleteAliasオペレーションを使用します。エイリアスを削除しても、基になる KMS キーに影響はありません。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$aliasName = "alias/projectKey1";

try {
    $result = $KmsClient->deleteAlias([
        'AliasName' => $aliasName,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS SDK for PHP バージョン 3 を使用した Amazon Kinesis のコード例

Amazon Kinesis は、データをリアルタイムで収集、処理、分析する AWS サービスです。Amazon Kinesis Data Streams を使用してデータストリームを設定するか、Amazon Data Firehose を使用して Amazon S3、 OpenSearch Service、Amazon Redshift、Splunk にデータを送信します。

Kinesis の詳細については、Amazon Kinesis のドキュメントを参照してください。

AWS SDK for PHP バージョン 3 のすべてのサンプルコードは、 にあります GitHub。

トピック

- Kinesis Data Streams API と AWS SDK for PHP バージョン 3 を使用したデータストリームの作成
- Kinesis Data Streams API と AWS SDK for PHP バージョン 3 を使用したデータシャードの管理
- Firehose API とAWS SDK for PHPバージョン 3 を使用した配信ストリームの作成

Kinesis Data Streams API と AWS SDK for PHP バージョン 3 を使用したデータストリームの作成

Amazon Kinesis Data Streams では、リアルタイムデータを送信できます。Kinesis Data Streams を使用して、データを追加するたびに設定された送信先にデータを配信するデータプロデューサーを作成します。

詳細については、「Amazon Kinesis デベロッパーガイド」の「ストリームの作成と管理」を参照してください。

以下の例では、次の方法を示しています。

- を使用してデータストリームを作成しますCreateAlias。
- を使用して、単一のデータストリームの詳細を取得しますDescribeStream。

- を使用して既存のデータストリームを一覧表示しますListStreams。
- を使用して既存のデータストリームにデータを送信しますPutRecord。
- を使用してデータストリームを削除しますDeleteStream。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

「Amazon Kinesis デベロッパーガイド」の使用に関する詳細については、「Amazon Kinesis Data Streams デベロッパーガイド」を参照してください。

Kinesis Data Streams を使用したデータストリームの作成

次のコード例を使用して、Kinesis によって処理される情報を送信できる Kinesis データストリームを確立します。「Amazon Kinesis デベロッパーガイド」でデータストリームの作成および更新の詳細について参照してください。

Kinesis データストリームを作成するには、 CreateStreamオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

$shardCount = 2;
$name = "my_stream_name";
```

```
try {
    $result = $kinesisClient->createStream([
        'ShardCount' => $shardCount,
        'StreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

データストリームを取得する

次のコード例を使用して、既存のデータストリームの詳細を取得します。デフォルトでは、これにより、指定した Kinesis データストリームに接続されている最初の 10 個のシャードに関する情報が返されます。必ず、Kinesis データストリームにデータを書き込む前に応答から `StreamStatus` を確認してください。

指定された Kinesis データストリームの詳細を取得するには、 DescribeStreamオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";

try {
    $result = $kinesisClient->describeStream([
```

```
        'StreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Kinesis に接続された既存のデータストリームのリスト

選択した AWS リージョンの AWS アカウント から最初の 10 個のデータストリームをリストします。返された `HasMoreStreams を使用して、さらに多くのストリームがアカウントに関連付けられているかどうかを判断します。

Kinesis データストリームを一覧表示するには、 ListStreamsオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

try {
    $result = $kinesisClient->listStreams();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

既存のデータストリームにデータを送信する

データストリームを作成したら、次の例を使用してデータを送信します。データを送信する前に、StreamStatus を使用してデータ DescribeStream がアクティブであるかどうかを確認します。

1 つのデータレコードを Kinesis データストリームに書き込むには、 PutRecordオペレーションを使用します。Kinesis データストリームに最大 500 個のレコードを書き込むには、 PutRecordsオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-1'
]);

$name = "my_stream_name";
$content = '{"ticker_symbol":"QXZ", "sector":"HEALTHCARE", "change":-0.05,
 "price":84.51}';
$groupID = "input to a hash function that maps the partition key (and associated data)
 to a specific shard";

try {
    $result = $kinesisClient->PutRecord([
        'Data' => $content,
        'StreamName' => $name,
        'PartitionKey' => $groupID
    ]);
    print("<p>ShardID = " . $result["ShardId"] . "</p>");
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
```

```
    echo "\n";
}
```

データストリームを削除する

この例では、データストリームを削除する方法を示しています。データストリームを削除すると、データストリームに送信されたデータもすべて削除されます。アクティブな Kinesis データストリームは、ストリームの削除が完了するまで DELETING 状態に切り替わります。DELETING 状態の間、ストリームはデータの処理を続けます。

Kinesis データストリームを削除するには、 DeleteStreamオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";

try {
    $result = $kinesisClient->deleteStream([
        'StreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Kinesis Data Streams API と AWS SDK for PHP バージョン 3 を使用したデータシャードの管理

Amazon Kinesis Data Streams では、エンドポイントにリアルタイムデータを送信できます。データフローのレートは、ストリーム内のシャードの数によって異なります。

1 つのシャードに 1 秒あたり 1,000 個のレコードを書き込むことができます。各シャードには、1 秒あたり 1 MiB のアップロード制限もあります。使用量は、シャード単位で計算および請求されるため、これらの例を使用してストリームのデータ容量とコストを管理します。

以下の例では、次の方法を示しています。

- を使用してストリーム内のシャードを一覧表示しますListShards。
- を使用して、ストリーム内のシャード数を追加または減らしますUpdateShardCount。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Amazon Kinesis Data Streams の使用の詳細については、「Amazon Kinesis Data Streams デベロッパーガイド」を参照してください。

データストリームシャードをリストする

特定のストリーム内にある最大 100 個のシャードの詳細をリストします。

Kinesis データストリーム内のシャードを一覧表示するには、 ListShardsオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";

try {
    $result = $kinesisClient->ListShards([
        'StreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

データストリームシャードをさらに追加する

データストリームシャードをさらに追加する必要がある場合、現在のシャード数を増やすことができます。増やす場合、シャード数を 2 倍にすることをお勧めします。これにより、現在利用可能な各シャードのコピーが作成され、容量が増加します。シャードの数を 2 倍にできるのは、24 時間以内に 2 回のみです。

先ほど説明したように Kinesis Data Streams の使用量の請求はシャードごとに計算されるため、需要が減ったらシャード数を半分に減らすことをお勧めします。シャードを削除するとき、シャードの量のみ現在のシャード数の半分に減らすことができます。

Kinesis データストリームのシャード数を更新するには、 UpdateShardCountオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";
$totalshards = 4;

try {
    $result = $kinesisClient->UpdateShardCount([
        'ScalingType' => 'UNIFORM_SCALING',
        'StreamName' => $name,
        'TargetShardCount' => $totalshards
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Firehose API とAWS SDK for PHPバージョン 3 を使用した配信ストリームの作成

Amazon Data Firehose を使用すると、Amazon Kinesis Data Streams、Amazon S3、Amazon OpenSearch Service (OpenSearch サービス) 、Amazon Redshift などの他の AWSのサービス、または Splunk にリアルタイムデータを送信できます。配信ストリームを持つデータプロデューサーを作成し、データを追加するたびに、設定された送信先にデータを配信します。

以下の例では、次の方法を示しています。

- を使用して配信ストリームを作成しますCreateDeliveryStream。
- を使用して 1 つの配信ストリームの詳細を取得しますDescribeDeliveryStream。
- を使用して配信ストリームを一覧表示しますListDeliveryStreams。
- を使用して配信ストリームにデータを送信しますPutRecord。
- を使用して配信ストリームを削除しますDeleteDeliveryStream。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Amazon Data Firehose の使用の詳細については、「Amazon Kinesis Data Firehose デベロッパーガイド」を参照してください。

Kinesis データストリームを使用した配信ストリームの作成

既存の Kinesis データストリームにデータを配置する配信ストリームを確立するには、CreateDeliveryStreamオペレーションを使用します。

これにより、デベロッパーは既存の Kinesis サービスを Firehose に移行できます。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";
$stream_type = "KinesisStreamAsSource";
$kinesis_stream = "arn:aws:kinesis:us-east-2:0123456789:stream/my_stream_name";
$role = "arn:aws:iam::0123456789:policy/Role";

try {
    $result = $firehoseClient->createDeliveryStream([
        'DeliveryStreamName' => $name,
        'DeliveryStreamType' => $stream_type,
        'KinesisStreamSourceConfiguration' => [
            'KinesisStreamARN' => $kinesis_stream,
            'RoleARN' => $role,
```

```
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Amazon S3 バケットを使用した配信ストリームの作成

既存の Amazon S3CreateDeliveryStream バケットにデータを配置する配信ストリームを確立するには、 オペレーションを使用します。

「Destination Parameters」で説明されているように、送信先パラメータを指定します。次に、「Amazon S3 の送信先 へのアクセス権を Kinesis Data Firehose に付与する」で説明されているように、Amazon S3 バケットへのアクセス権を Firehose に付与するようにしてください。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_S3_stream_name";
$stream_type = "DirectPut";
$s3bucket = 'arn:aws:s3:::bucket_name';
$s3Role = 'arn:aws:iam::0123456789:policy/Role';

try {
    $result = $firehoseClient->createDeliveryStream([
        'DeliveryStreamName' => $name,
```

```
        'DeliveryStreamType' => $stream_type,
        'S3DestinationConfiguration' => [
            'BucketARN' => $s3bucket,
            'CloudWatchLoggingOptions' => [
                'Enabled' => false,
            ],
            'RoleARN' => $s3Role
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

OpenSearch サービスを使用して配信ストリームを作成する

OpenSearch サービスクラスターにデータを配置する Firehose 配信ストリームを確立するには、CreateDeliveryStreamオペレーションを使用します。

「Destination Parameters」で説明されているように、送信先パラメータを指定します。「Amazon ES 送信先 へのアクセス権を Kinesis Data Firehose に付与する」の説明に従って、OpenSearch サービスクラスターへのアクセス権を Firehose に付与してください。 https://docs.aws.amazon.com/firehose/latest/dev/controlling-access.html#using-iam-es.html

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);
```

```
$name = "my_ES_stream_name";
$stream_type = "DirectPut";
$esDomainARN = 'arn:aws:es:us-east-2:0123456789:domain/Name';
$esRole = 'arn:aws:iam::0123456789:policy/Role';
$esIndex = 'root';
$esType = 'PHP_SDK';
$s3bucket = 'arn:aws:s3:::bucket_name';
$s3Role = 'arn:aws:iam::0123456789:policy/Role';

try {
    $result = $firehoseClient->createDeliveryStream([
        'DeliveryStreamName' => $name,
        'DeliveryStreamType' => $stream_type,
        'ElasticsearchDestinationConfiguration' => [
            'DomainARN' => $esDomainARN,
            'IndexName' => $esIndex,
            'RoleARN' => $esRole,
            'S3Configuration' => [
                'BucketARN' => $s3bucket,
                'CloudWatchLoggingOptions' => [
                    'Enabled' => false,
                ],
                'RoleARN' => $s3Role,
            ],
            'TypeName' => $esType,
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

配信ストリームを取得する

既存の Firehose 配信ストリームの詳細を取得するには、 DescribeDeliveryStreamオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";

try {
    $result = $firehoseClient->describeDeliveryStream([
        'DeliveryStreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Kinesis Data Streams に接続された既存の配信ストリームのリスト

Kinesis Data Streams にデータを送信する既存の Firehose 配信ストリームをすべて一覧表示するには、 ListDeliveryStreamsオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
```

```
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

try {
    $result = $firehoseClient->listDeliveryStreams([
        'DeliveryStreamType' => 'KinesisStreamAsSource',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

他の AWS サービスにデータを送信している既存の配信ストリームをリストする

Amazon S3、Service、または Amazon Redshift、または Splunk にデータを送信する既存の Firehose 配信ストリームをすべて一覧表示するには、 ListDeliveryStreamsオペレーションを使用します。 OpenSearch

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

try {
    $result = $firehoseClient->listDeliveryStreams([
        'DeliveryStreamType' => 'DirectPut',
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

既存の Firehose 配信ストリームにデータを送信する

Firehose 配信ストリームを介して指定した送信先にデータを送信するには、Firehose 配信ストリームを作成した後に PutRecordオペレーションを使用します。

Firehose 配信ストリームにデータを送信する前に、 DescribeDeliveryStreamを使用して配信ストリームがアクティブかどうかを確認します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";
$content = '{"ticker_symbol":"QXZ", "sector":"HEALTHCARE", "change":-0.05,
 "price":84.51}';

try {
    $result = $firehoseClient->putRecord([
        'DeliveryStreamName' => $name,
        'Record' => [
            'Data' => $content,
        ],
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Firehose 配信ストリームを削除する

Firehose 配信ストリームを削除するには、 DeleteDeliveryStreamsオペレーションを使用します。これにより、送信ストリームに送信したデータがすべて削除されます。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";

try {
    $result = $firehoseClient->deleteDeliveryStream([
        'DeliveryStreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS Elemental MediaConvertAWS SDK for PHP バージョン 3 を使用したの例

AWS Elemental MediaConvert は、ブロードキャストグレード機能を備えたファイルベースのビデオトランスコードサービスです。これを使用して、インターネット経由でブロードキャストおよび (VOD) 配信用の video-on-demandアセットを作成できます。詳細については、『AWS Elemental MediaConvert ユーザーガイド』を参照してください。

の PHP API AWS Elemental MediaConvert は、 *`AWS.MediaConvert`* クライアントクラスを通じて公開されます。詳細については、API リファレンスの`Class: AWS.MediaConvert`を参照してください。

でのトランスコードジョブの作成と管理 AWS Elemental MediaConvert

この例では、 AWS SDK for PHP バージョン 3 を使用して を呼び出し AWS Elemental MediaConvert 、トランスコードジョブを作成します。開始する前に、入力ストレージとしてプロビジョニングした Amazon S3 バケットに対して、入力動画をアップロードする必要があります。サポートされている入力動画のコーデックとコンテナの一覧については、「AWS Elemental MediaConvert ユーザーガイド」の「サポートされる入力コーデックおよびコンテナ」を参照してください。

以下の例では、次の方法を示しています。

- でトランスコードジョブを作成します AWS Elemental MediaConvert。CreateJob。
- AWS Elemental MediaConvert キューからトランスコードジョブをキャンセルします。 CancelJob
- 完了したトランスコードジョブの JSON を取得します。 GetJob
- 最後に作成されたジョブのうち最大 20 個の JSON 配列を取得します。 ListJobs

のすべてのサンプルコード AWS SDK for PHP は、 で GitHub入手できます。

認証情報

サンプルコードを実行する前に、「」の説明に従って AWS 認証情報を設定します認証情報。次に AWS SDK for PHP、「」の説明に従って をインポートします基本的な使用法。

MediaConvert クライアントにアクセスするには、入力ファイルと出力ファイルが保存されている Amazon S3 バケット AWS Elemental MediaConvert へのアクセスを許可する IAM ロールを作成します。詳細については、「AWS Elemental MediaConvert ユーザーガイド」の「IAM アクセス許可の設定」を参照してください。

クライアントの作成

コードのリージョンで MediaConvert クライアント AWS SDK for PHP を作成して、 を設定します。次の例では、リージョンを us-west-2 に設定しています。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\MediaConvert\MediaConvertClient;
```

サンプルコード

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);
```

シンプルなコード変換ジョブの定義

コード変換ジョブのパラメータを定義する JSON を作成します。

これらは詳細パラメータです。AWS Elemental MediaConvert コンソールを使用して JSON ジョブのパラメータを生成できます。そのためには、コンソールでジョブ設定を選択し、[ジョブ] セクションの下部にある [ジョブ JSON の表示] を選択します。次の例は、シンプルなジョブの JSON を示しています。

サンプルコード

```
$jobSetting = [
    "OutputGroups" => [
        [
            "Name" => "File Group",
            "OutputGroupSettings" => [
                "Type" => "FILE_GROUP_SETTINGS",
                "FileGroupSettings" => [
                    "Destination" => "s3://OUTPUT_BUCKET_NAME/"
                ]
            ],
```

```
            "Outputs" => [
                [
                    "VideoDescription" => [
                        "ScalingBehavior" => "DEFAULT",
                        "TimecodeInsertion" => "DISABLED",
                        "AntiAlias" => "ENABLED",
                        "Sharpness" => 50,
                        "CodecSettings" => [
                            "Codec" => "H_264",
                            "H264Settings" => [
                                "InterlaceMode" => "PROGRESSIVE",
                                "NumberReferenceFrames" => 3,
                                "Syntax" => "DEFAULT",
                                "Softness" => 0,
                                "GopClosedCadence" => 1,
                                "GopSize" => 90,
                                "Slices" => 1,
                                "GopBReference" => "DISABLED",
                                "SlowPal" => "DISABLED",
                                "SpatialAdaptiveQuantization" => "ENABLED",
                                "TemporalAdaptiveQuantization" => "ENABLED",
                                "FlickerAdaptiveQuantization" => "DISABLED",
                                "EntropyEncoding" => "CABAC",
                                "Bitrate" => 5000000,
                                "FramerateControl" => "SPECIFIED",
                                "RateControlMode" => "CBR",
                                "CodecProfile" => "MAIN",
                                "Telecine" => "NONE",
                                "MinIInterval" => 0,
                                "AdaptiveQuantization" => "HIGH",
                                "CodecLevel" => "AUTO",
                                "FieldEncoding" => "PAFF",
                                "SceneChangeDetect" => "ENABLED",
                                "QualityTuningLevel" => "SINGLE_PASS",
                                "FramerateConversionAlgorithm" => "DUPLICATE_DROP",
                                "UnregisteredSeiTimecode" => "DISABLED",
                                "GopSizeUnits" => "FRAMES",
                                "ParControl" => "SPECIFIED",
                                "NumberBFramesBetweenReferenceFrames" => 2,
                                "RepeatPps" => "DISABLED",
                                "FramerateNumerator" => 30,
                                "FramerateDenominator" => 1,
                                "ParNumerator" => 1,
                                "ParDenominator" => 1
```

```
                        ]
                    ],
                    "AfdSignaling" => "NONE",
                    "DropFrameTimecode" => "ENABLED",
                    "RespondToAfd" => "NONE",
                    "ColorMetadata" => "INSERT"
                ],
                "AudioDescriptions" => [
                    [
                        "AudioTypeControl" => "FOLLOW_INPUT",
                        "CodecSettings" => [
                            "Codec" => "AAC",
                            "AacSettings" => [
                                "AudioDescriptionBroadcasterMix" => "NORMAL",
                                "RateControlMode" => "CBR",
                                "CodecProfile" => "LC",
                                "CodingMode" => "CODING_MODE_2_0",
                                "RawFormat" => "NONE",
                                "SampleRate" => 48000,
                                "Specification" => "MPEG4",
                                "Bitrate" => 64000
                            ]
                        ],
                        "LanguageCodeControl" => "FOLLOW_INPUT",
                        "AudioSourceName" => "Audio Selector 1"
                    ]
                ],
                "ContainerSettings" => [
                    "Container" => "MP4",
                    "Mp4Settings" => [
                        "CslgAtom" => "INCLUDE",
                        "FreeSpaceBox" => "EXCLUDE",
                        "MoovPlacement" => "PROGRESSIVE_DOWNLOAD"
                    ]
                ],
                "NameModifier" => "_1"
            ]
        ]
    ]
],
"AdAvailOffset" => 0,
"Inputs" => [
    [
        "AudioSelectors" => [
```

```
                "Audio Selector 1" => [
                    "Offset" => 0,
                    "DefaultSelection" => "NOT_DEFAULT",
                    "ProgramSelection" => 1,
                    "SelectorType" => "TRACK",
                    "Tracks" => [
                        1
                    ]
                ]
            ],
            "VideoSelector" => [
                "ColorSpace" => "FOLLOW"
            ],
            "FilterEnable" => "AUTO",
            "PsiControl" => "USE_PSI",
            "FilterStrength" => 0,
            "DeblockFilter" => "DISABLED",
            "DenoiseFilter" => "DISABLED",
            "TimecodeSource" => "EMBEDDED",
            "FileInput" => "s3://INPUT_BUCKET_AND_FILE_NAME"
        ]
    ],
    "TimecodeConfig" => [
        "Source" => "EMBEDDED"
    ]
];
```

ジョブの作成

ジョブパラメータの JSON を作成した後で、AWS.MediaConvert service object を呼び出して createJob メソッドを呼び出し、パラメータを渡します。作成されたジョブの ID がレスポンスのデータで返されます。

サンプルコード

```
try {
    $result = $mediaConvertClient->createJob([
        "Role" => "IAM_ROLE_ARN",
        "Settings" => $jobSetting, //JobSettings structure
        "Queue" => "JOB_QUEUE_ARN",
        "UserMetadata" => [
            "Customer" => "Amazon"
        ],
```

```
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

ジョブの取得

createjob を呼び出して返された JobID を使用して、最近のジョブの詳細な説明を JSON 形式で取得できます。

サンプルコード

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);

try {
    $result = $mediaConvertClient->getJob([
        'Id' => 'JOB_ID',
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

ジョブのキャンセル

createjob を呼び出して返された JobID を使用して、キュー内にあるジョブをキャンセルできます。コード変換を既に開始しているジョブをキャンセルすることはできません。

サンプルコード

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
```

```
    'profile' => 'default'
]);

try {
    $result = $mediaConvertClient->cancelJob([
        'Id' => 'JOB_ID', // REQUIRED The Job ID of the job to be cancelled.
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

最新のコード変換ジョブの一覧表示

パラメータの JSON を作成します。これには、リストを昇順または降順のいずれでソートするかを指定する値、チェックするジョブキューの ARN、および追加するジョブのステータスが含まれます。最大 20 個のジョブが返されます。次の 20 個の最新ジョブを取得するには、結果で返される nextToken 文字列を使用します。

サンプルコード

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);

try {
    $result = $mediaConvertClient->listJobs([
        'MaxResults' => 20,
        'Order' => 'ASCENDING',
        'Queue' => 'QUEUE_ARN',
        'Status' => 'SUBMITTED',
        // 'NextToken' => '<string>', //OPTIONAL To retrieve the twenty next most
 recent jobs
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS SDK for PHP バージョン 3 を使用した Amazon S3 のコードサンプル

Amazon Simple Storage Service (Amazon S3) は、拡張性の高いクラウドストレージを提供するウェブサービスです。Amazon S3 は簡単に使用できるオブジェクトストレージです。シンプルなウェブサービスインターフェイスが用意されており、ウェブ上のどこからでも容量に関係なくデータを保存、取得できます。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

トピック

- AWS SDK for PHP バージョン 3 での Amazon S3 バケットの作成と使用
- AWS SDK for PHP バージョン 3 による Amazon S3 バケットへのアクセス許可の管理
- AWS SDK for PHP バージョン 3 による Amazon S3 バケットの設定
- AWS SDK for PHP バージョン 3 での Amazon S3 マルチパートアップロードの使用
- AWS SDK for PHP バージョン 3 を使用した Amazon S3 の署名付き URL
- AWS SDK for PHP バージョン 3 での Amazon S3 の署名付き POST
- AWS SDK for PHP バージョン 3 での静的ウェブホストとしての Amazon S3 バケットの使用
- AWS SDK for PHP バージョン 3 での Amazon S3 バケットポリシーの使用
- AWS SDK for PHP バージョン 3 での S3 アクセスポイント ARN の使用
- AWS SDK for PHP バージョン 3 で Amazon S3 マルチリージョンアクセスポイントを使用する

AWS SDK for PHP バージョン 3 での Amazon S3 バケットの作成と使用

以下の例では、次の方法を示しています。

- を使用して、リクエストの認証済み送信者が所有するバケットのリストを返しますListBuckets。
- を使用して新しいバケットを作成しますCreateBucket。
- を使用してバケットにオブジェクトを追加しますPutObject。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

インポート

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
```

バケットの一覧表示

次のコードのように、PHP ファイルを作成します。まず AWS リージョンとバージョンを指定する AWS.S3 クライアントサービスを作成します。次に `listBuckets` メソッドを呼び出します。これはリクエストの送信者が所有しているすべての Amazon S3 バケットをバケット配列として返します。

サンプルコード

```
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

//Listing all S3 Bucket
$buckets = $s3Client->listBuckets();
foreach ($buckets['Buckets'] as $bucket) {
    echo $bucket['Name'] . "\n";
}
```

バケットを作成する

次のコードのように、PHP ファイルを作成します。まず AWS リージョンとバージョンを指定する AWS.S3 クライアントサービスを作成します。次に、`createBucket` メソッドを、パラメーターの

配列を使用して呼び出します。唯一の必須のフィールドはキー 'Bucket' で、作成するバケット名の文字列値を指定します。ただし、CreateBucketConfiguration「」フィールドを使用してAWSリージョンを指定できます。成功した場合、このメソッドはバケットの「Location」を返します。

サンプルコード

```
function createBucket($s3Client, $bucketName)
{
    try {
        $result = $s3Client->createBucket([
            'Bucket' => $bucketName,
        ]);
        return 'The bucket\'s location is: ' .
            $result['Location'] . '. ' .
            'The bucket\'s effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function createTheBucket()
{
    $s3Client = new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2006-03-01'
    ]);

    echo createBucket($s3Client, 'my-bucket');
}

// Uncomment the following line to run this code in an AWS account.
// createTheBucket();
```

バケットにオブジェクトを配置する

新しいバケットにファイルを追加するには、次のコードを使用して PHP ファイルを作成します。

コマンドラインでこのファイルを実行し、ファイルをアップロードするバケットの名前を文字列、続いてアップロードするファイルの完全なファイルパスを渡します。

サンプルコード

```
$USAGE = "\n" .
    "To run this example, supply the name of an S3 bucket and a file to\n" .
    "upload to it.\n" .
    "\n" .
    "Ex: php PutObject.php <bucketname> <filename>\n";

if (count($argv) <= 2) {
    echo $USAGE;
    exit();
}

$bucket = $argv[1];
$file_Path = $argv[2];
$key = basename($argv[2]);

try {
    //Create a S3Client
    $s3Client = new S3Client([
        'profile' => 'default',
        'region' => 'us-west-2',
        'version' => '2006-03-01'
    ]);
    $result = $s3Client->putObject([
        'Bucket' => $bucket,
        'Key' => $key,
        'SourceFile' => $file_Path,
    ]);
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

AWS SDK for PHP バージョン 3 による Amazon S3 バケットへのアクセス許可の管理

アクセスコントロールリスト (ACL) はリソースベースのアクセスポリシーオプションの 1 つであり、バケットとオブジェクトへのアクセスを管理するために使用できます。ACL を使用して、基本的な読み取り/書き込みアクセス許可を他の AWS アカウントに付与できます。詳細については、「ACL によるアクセス管理」を参照してください。

以下の例では、次の方法を示しています。

- GetBucketAcl を使用した、バケットに対するアクセスコントロールポリシーの取得。

- ACL、PutBucketAcl を使用した、バケットのアクセス許可の設定。

AWS SDK for PHP 用のすべてのサンプルコードは GitHub で入手できます。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

アクセスコントロールリストポリシーの取得と設定

インポート

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;
```

サンプルコード

```

// Create a S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

// Gets the access control policy for a bucket
$bucket = 'my-s3-bucket';
try {
    $resp = $s3Client->getBucketAcl([
        'Bucket' => $bucket
    ]);
    echo "Succeed in retrieving bucket ACL as follows: \n";
    var_dump($resp);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

// Sets the permissions on a bucket using access control lists (ACL).
```

```
$params = [
    'ACL' => 'public-read',
    'AccessControlPolicy' => [
        // Information can be retrieved from `getBucketAcl` response
        'Grants' => [
            [
                'Grantee' => [
                    'DisplayName' => '<string>',
                    'EmailAddress' => '<string>',
                    'ID' => '<string>',
                    'Type' => 'CanonicalUser',
                    'URI' => '<string>',
                ],
                'Permission' => 'FULL_CONTROL',
            ],
            // ...
        ],
        'Owner' => [
            'DisplayName' => '<string>',
            'ID' => '<string>',
        ],
    ],
    'Bucket' => $bucket,
];

try {
    $resp = $s3Client->putBucketAcl($params);
    echo "Succeed in setting bucket ACL.\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}
```

AWS SDK for PHP バージョン 3 による Amazon S3 バケットの設定

Cross−Origin Resource Sharing (CORS) は、特定のドメインにロードされたクライアントウェブアプリケーションが異なるドメイン内のリソースと通信する方法を定義します。Amazon S3 の CORS のサポートによって、Amazon S3 でリッチなクライアント側ウェブアプリケーションを構築し、Amazon S3 リソースへのクロスオリジンアクセスを選択的に許可できます。

Amazon S3 バケットで CORS 設定を使用する方法の詳細については、「Cross-Origin Resource Sharing (CORS)」を参照してください。

以下の例では、次の方法を示しています。

- を使用してバケットの CORS 設定を取得しますGetBucketCors。
- を使用してバケットの CORS 設定を設定しますPutBucketCors。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

CORS 設定を取得する

次のコードのように、PHP ファイルを作成します。AWS.S3 クライアントサービスを作成してから、`getBucketCors` メソッドを呼び出して、CORS 設定を取得するバケットを指定します。

必要なパラメーターは、選択したバケットの名前のみです。そのバケットに CORS 設定が存在する場合は、その設定が Amazon S3 によって CORSRules オブジェクトとして返されます。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\S3\S3Client;
```

サンプルコード

```
$client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

try {
```

```
    $result = $client->getBucketCors([
        'Bucket' => $bucketName, // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

CORS 設定を設定する

次のコードのように、PHP ファイルを作成します。AWS.S3 クライアントサービスを作成してから、putBucketCors メソッドを呼び出して、CORS 設定を設定するバケットを指定し、CORSConfiguration を CORSRules JSON オブジェクトとして指定します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\S3\S3Client;
```

サンプルコード

```
$client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

try {
    $result = $client->putBucketCors([
        'Bucket' => $bucketName, // REQUIRED
        'CORSConfiguration' => [ // REQUIRED
            'CORSRules' => [ // REQUIRED
                [
                    'AllowedHeaders' => ['Authorization'],
                    'AllowedMethods' => ['POST', 'GET', 'PUT'], // REQUIRED
                    'AllowedOrigins' => ['*'], // REQUIRED
```

```
                    'ExposeHeaders' => [],
                    'MaxAgeSeconds' => 3000
                ],
            ],
        ]
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 での Amazon S3 マルチパートアップロードの使用

1 回の PutObject 操作では、合計サイズが 5 GB 以内のオブジェクトを
アップロードできます。ただし、マルチパートアップロード手法 (たとえ
ば、CreateMultipartUpload、UploadPart、CompleteMultipartUpload、AbortMultipartUplo
を使用すると、合計サイズが 5 MB ~ 5 TB のオブジェクトをアップロードできます。

以下の例では、次の方法を示しています。

- を使用して Amazon S3 にオブジェクトをアップロードしますObjectUploader。
- を使用して Amazon S3 オブジェクトのマルチパートアップロードを作成しま
  すMultipartUploader。
- を使用して、ある Amazon S3 の場所から別の場所にオブジェクトをコピーしますObjectCopier。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

オブジェクトアップローダー

PutObject または MultipartUploader がタスクに最適かどうかが不明な場合
は、ObjectUploader を使用します。ObjectUploader は、ペイロードサイズに基づいてどれが
最適かにより、PutObject または MultipartUploader を使用して大きなファイルを Amazon S3
にアップロードします。

```
require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\ObjectUploader;
use Aws\S3\S3Client;
```

サンプルコード

```
// Create an S3Client.
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-east-2',
    'version' => '2006-03-01'
]);

$bucket = 'your-bucket';
$key = 'my-file.zip';

// Use a stream instead of a file path.
$source = fopen('/path/to/large/file.zip', 'rb');

$uploader = new ObjectUploader(
    $s3Client,
    $bucket,
    $key,
    $source
);

do {
    try {
        $result = $uploader->upload();
        if ($result["@metadata"]["statusCode"] == '200') {
            print('<p>File successfully uploaded to ' . $result["ObjectURL"] . '.</
p>');
        }
        print($result);
        // If the SDK chooses a multipart upload, try again if there is an exception.
        // Unlike PutObject calls, multipart upload calls are not automatically
 retried.
    } catch (MultipartUploadException $e) {
        rewind($source);
        $uploader = new MultipartUploader($s3Client, $source, [
```

```
            'state' => $e->getState(),
        ]);
    }
} while (!isset($result));

fclose($source);
```

構成

ObjectUploader オブジェクトのコンストラクタでは次の引数を指定できます。

$client

転送の実行に使用する Aws\ClientInterface オブジェクト。これは Aws\S3\S3Client のインスタンスである必要があります。

$bucket

(string、必須) オブジェクトのアップロード先のバケットの名前。

$key

(string、必須) アップロードするオブジェクトで使用するキー。

$body

(mixed、必須) アップロードするオブジェクトデータ。StreamInterface、PHP ストリームリソース、またはアップロードするデータ文字列のいずれでもかまいません。

$acl

(string) をアップロードするオブジェクトに設定するアクセスコントロールリスト (ACL)。デフォルトでは、オブジェクトはプライベートです。

$options

マルチパートアップロードの設定オプションの連想配列。有効な設定オプションは次のとおりです。

add_content_md5

(bool) true に設定すると、アップロードの MD5 チェックサムが自動的に計算されます。

mup_threshold

(int、デフォルト:int(16777216)) ファイルサイズのバイト数。ファイルサイズがこの制限を超えると、マルチパートアップロードが使用されます。

before_complete

(callable) CompleteMultipartUpload オペレーションの前に呼び出すコールバック。このコールバックは、function (Aws\Command $command) {...} のような関数の署名を持っている必要があります。

before_initiate

(callable) CreateMultipartUpload オペレーションの前に呼び出すコールバック。このコールバックは、function (Aws\Command $command) {...} のような関数の署名を持っている必要があります。

before_upload

(callable) すべての UploadPart または PutObject オペレーションの前に呼び出すコールバック。このコールバックは、function (Aws\Command $command) {...} のような関数の署名を持っている必要があります。

concurrency

(int、デフォルト: int(3)) マルチパートアップロード中に許容される同時 UploadPart オペレーションの最大数。

part_size

(int、デフォルト: int(5242880)) マルチパートアップロードの実行時に使用するパートサイズ (バイト単位)。値は、5 MB 以上かつ 5 GB 以内である必要があります。

state

(Aws\Multipart\UploadState) マルチパートアップロードの状態を表すオブジェクトであり、前回のアップロードを再開するために使用されます。このオプションを指定している場合、$bucket および $key 引数と part_size オプションは無視されます。

MultipartUploader

マルチパートアップロードは、大容量オブジェクトのアップロードを効率よく行えるように設計されています。マルチパートアップロードでは、オブジェクトを分割して、別々に任意の順序で並行してアップロードできます。

Amazon S3 のユーザーには、100 MB を超えるオブジェクトに対してマルチパートアップロードを使用することをお勧めします。

MultipartUploader オブジェクト

SDK には、マルチパートアップロードのプロセスを簡素化する特別な `MultipartUploader` オブジェクトがあります。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

サンプルコード

```
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

// Use multipart upload
$source = '/path/to/large/file.zip';
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);

try {
    $result = $uploader->upload();
    echo "Upload complete: {$result['ObjectURL']}\n";
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . "\n";
}
```

このアップローダーでは、指定されたソースと設定に基づいてパートデータのジェネレーターが作成され、すべてのパートのアップロードが試行されます。一部のパートアップロードが失敗すると、アップローダーによってソースデータ全体が読み取られるまで、失敗したパートのアップロードが続行されます。その後、アップローダーは失敗したパートのアップロードを再試行するか、アップロードに失敗したパートに関する情報を含む例外をスローします。

マルチパートアップロードのカスタマイズ

マルチパートアップローダーによって実行される CreateMultipartUpload、UploadPart、CompleteMultipartUpload オペレーションに対して、コンストラクタに渡すコールバックを介してカスタムオプションを設定できます。

インポート

```
require 'vendor/autoload.php';

use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

サンプルコード

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

// Customizing a multipart upload
$source = '/path/to/large/file.zip';
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
    'before_initiate' => function (Command $command) {
        // $command is a CreateMultipartUpload operation
        $command['CacheControl'] = 'max-age=3600';
    },
    'before_upload' => function (Command $command) {
        // $command is an UploadPart operation
        $command['RequestPayer'] = 'requester';
    },
    'before_complete' => function (Command $command) {
        // $command is a CompleteMultipartUpload operation
        $command['RequestPayer'] = 'requester';
    },
]);
```

パートのアップロード間の手動のガベージコレクション

大きなアップロードでメモリ制限に達している場合、メモリ制限に達したときに PHP ガベージコレクターで収集されていない SDK によって生成された巡回参照が原因である可能性があります。オペレーション間で収集アルゴリズムを手動で呼び出すと、制限に達する前にサイクルを収集できます。次の例では、各パートをアップロードする前にコールバックを使用して、収集アルゴリズムを呼び出します。ガベージコレクターを呼び出してもパフォーマンスに影響することはなく、最適な使用はお客様のユースケースと環境によって異なることに注意してください。

```
$uploader = new MultipartUploader($client, $source, [
   'bucket' => 'your-bucket',
   'key' => 'your-key',
   'before_upload' => function(\Aws\Command $command) {
      gc_collect_cycles();
   }
]);
```

エラーからの復旧

マルチパートアップロードのプロセスでエラーが発生すると MultipartUploadException がスローされます。この例外では、マルチパートアップロードの進行状況に関する情報が含まれている UploadState オブジェクトへのアクセスが提供されます。UploadState を使用して、完了できなかったアップロードを再開できます。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

サンプルコード

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
```

```
]);

$source = '/path/to/large/file.zip';
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);

//Recover from errors
do {
    try {
        $result = $uploader->upload();
    } catch (MultipartUploadException $e) {
        $uploader = new MultipartUploader($s3Client, $source, [
            'state' => $e->getState(),
        ]);
    }
} while (!isset($result));

//Abort a multipart upload if failed
try {
    $result = $uploader->upload();
} catch (MultipartUploadException $e) {
    // State contains the "Bucket", "Key", and "UploadId"
    $params = $e->getState()->getId();
    $result = $s3Client->abortMultipartUpload($params);
}
```

UploadState によるアップロードの再開では、まだアップロードされていないパートのアップロードが試行されます。この状態オブジェクトでは、パートが連続していない場合であっても、欠落しているパートが追跡されます。アップローダーは、指定されたソースファイルで、アップロードする必要があるパートに属するバイト範囲まで読み取りまたはシークします。

UploadState オブジェクトはシリアル化可能であるため、別のプロセスでアップロードを再開することもできます。また、例外を処理していない場合でも、UploadState を呼び出すことによって $uploader->getState() オブジェクトを取得できます。

> **Important**
>
> MultipartUploader にソースとして渡されるストリームは、アップロード前に自動的には巻き戻しされません。そのため、前述の例のようなループでファイルパスではなくストリームを使用している場合は、catch ブロック内で $source 変数をリセットします。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

サンプルコード

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

//Using stream instead of file path
$source = fopen('/path/to/large/file.zip', 'rb');
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);

do {
    try {
        $result = $uploader->upload();
    } catch (MultipartUploadException $e) {
        rewind($source);
        $uploader = new MultipartUploader($s3Client, $source, [
            'state' => $e->getState(),
        ]);
    }
} while (!isset($result));
```

```
fclose($source);
```

マルチパートアップロードの中止

マルチパートアップロードは、UploadState オブジェクトに含まれた UploadId を取得し、abortMultipartUpload に渡すことで中止できます。

```
try {
    $result = $uploader->upload();
} catch (MultipartUploadException $e) {
    // State contains the "Bucket", "Key", and "UploadId"
    $params = $e->getState()->getId();
    $result = $s3Client->abortMultipartUpload($params);
}
```

非同期マルチパートアップロード

upload() で MultipartUploader を呼び出すとリクエストがブロックされます。非同期コンテキストを使用している場合は、マルチパートアップロードの promise を取得できます。

```
require 'vendor/autoload.php';

use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

サンプルコード

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

$source = '/path/to/large/file.zip';
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);
```

```
$promise = $uploader->promise();
```

構成

MultipartUploader オブジェクトのコンストラクタでは次の引数を指定できます。

$client

転送の実行に使用する Aws\ClientInterface オブジェクト。これは Aws\S3\S3Client のインスタンスである必要があります。

$source

アップロードするソースデータ。これは、パスや URL (例: /path/to/file.jpg)、リソースハンドル (例: fopen('/path/to/file.jpg', 'r'))、PSR-7 ストリームのインスタンスのいずれかです。

$config

マルチパートアップロードの設定オプションの連想配列。

有効な設定オプションは次のとおりです。

acl

(string) をアップロードするオブジェクトに設定するアクセスコントロールリスト (ACL)。デフォルトでは、オブジェクトはプライベートです。

before_complete

(callable) CompleteMultipartUpload オペレーションの前に呼び出すコールバック。このコールバックは、function (Aws\Command $command) {...} のような関数の署名を持っている必要があります。

before_initiate

(callable) CreateMultipartUpload オペレーションの前に呼び出すコールバック。このコールバックは、function (Aws\Command $command) {...} のような関数の署名を持っている必要があります。

before_upload

(callable) すべての UploadPart オペレーションの前に呼び出すコールバック。このコールバックは、function (Aws\Command $command) {...} のような関数の署名を持っている必要があります。

bucket

(string、必須) オブジェクトのアップロード先のバケットの名前。

concurrency

(int、デフォルト: int(5)) マルチパートアップロード中に許容される同時 UploadPart オペレーションの最大数。

key

(string、必須) アップロードするオブジェクトで使用するキー。

part_size

(int、デフォルト: int(5242880)) マルチパートアップロードの実行時に使用するパートサイズ (バイト単位)。5 MB 以上かつ 5 GB 以内である必要があります。

state

(Aws\Multipart\UploadState) マルチパートアップロードの状態を表すオブジェクトであり、前回のアップロードを再開するために使用されます。このオプションを指定している場合、bucket、key、part_size オプションは無視されます。

add_content_md5

(boolean) true に設定すると、アップロードの MD5 チェックサムが自動的に計算されます。

マルチパートコピー

AWS SDK for PHP には、5 GB 以上かつ 5 TB 以内のサイズのオブジェクトを Amazon S3 内でコピーするように設計されている MultipartCopy オブジェクトも含まれていて、MultipartUploader と同様の方法で使用できます。

```
require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartCopy;
use Aws\S3\S3Client;
```

サンプルコード

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

//Copy objects within S3
$copier = new MultipartCopy($s3Client, '/bucket/key?versionId=foo', [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);

try {
    $result = $copier->copy();
    echo "Copy complete: {$result['ObjectURL']}\n";
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . "\n";
}
```

AWS SDK for PHP バージョン 3 を使用した Amazon S3 の署名付き URL

Authorization HTTP ヘッダーを使用する代わりに、必要な情報をクエリ文字列パラメーターとして渡すことで、特定の種類のリクエストを認証できます。これは、サードパーティーのブラウザで、リクエストのプロキシを行わずにプライベートの Amazon S3 データに直接アクセスさせる場合に便利です。これを行うには、「署名付き」のリクエストを作成し、エンドユーザーのブラウザが取得できる URL としてエンコードします。さらに、署名付きのリクエストは、有効期限を指定することで制限できます。

以下の例では、次の方法を示しています。

- を使用して S3 オブジェクトを取得するための署名付き URL を作成しますcreatePresignedRequest。

のすべてのサンプルコード AWS SDK for PHP は、 で GitHub入手できます。

認証情報

サンプルコードを実行する前に、「」の説明に従って AWS 認証情報を設定します認証情報。次に AWS SDK for PHP、「」の説明に従って をインポートします基本的な使用法。

署名付きリクエストの作成

Aws\S3\S3Client::createPresignedRequest() メソッドを使用して Amazon S3 オブジェクトへの署名付き URL を取得できます。このメソッドでは、Aws\CommandInterface オブジェクトと期限切れタイムスタンプを受け付け、署名付き Psr\Http\Message\RequestInterface オブジェクトを返します。リクエストの getUri() メソッドを使用して、オブジェクトの署名付き URL を取得できます。

最も一般的なシナリオは、オブジェクトを GET するために、署名付き URL を作成します。

インポート

```
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

require 'vendor/autoload.php';
```

サンプルコード

```
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
```

署名付き URL の作成

任意の Amazon S3 オペレーションに対する署名付き URL を作成できます。コマンドオブジェクトを作成するために getCommand メソッドを使用して、そのコマンドを使用して createPresignedRequest() メソッドを呼び出します。最終的にリクエストを送信するときは、返すリクエストと同じメソッドと同じヘッダーを必ず使用します。

サンプルコード

```
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the next
 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
```

```
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
```

オブジェクトの URL の取得

Amazon S3 バケットに保存されたオブジェクトへのパブリック URL のみが必要な場合は、`Aws\S3\S3Client::getObjectUrl()` メソッドを使用できます。このメソッドは、指定されたバケットとキーの署名なしの URL を返します。

サンプルコード

```
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
```

> **Important**
>
> このメソッドから返された URL はバケットまたはキーが必ず存在することを検証しません。またこのメソッドではオブジェクトが認証されていないアクセスを許可することも保証されません。

AWS SDK for PHP バージョン 3 での Amazon S3 の署名付き POST

署名付き URL と同様に、署名付き POST を使用すると、AWS 認証情報を付与しないでユーザーに書き込みアクセスを与えることができます。AwsS3PostObjectV4 のインスタンスを使用して、署名付き POST フォームを作成できます。

以下の例では、次の方法を示しています。

- PostObjectV4 を使用して、S3 オブジェクトの POST アップロード形式のデータを取得します。

AWS SDK for PHP 用のすべてのサンプルコードは GitHub で入手できます。

認証情報

> **Note**
>
> `PostObjectV4` は、AWS IAM Identity Centerからの認証情報では機能しません。

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

PostObjectV4 の作成

PostObjectV4 のインスタンスを作成するには、次を提供する必要があります。

- Aws\S3\S3Client のインスタンス
- bucket (バケット)
- フォーム入力フィールドの連想配列
- ポリシー条件の配列 (「Amazon Simple Storage Service ユーザーガイド」の「ポリシーの作成」を参照)。
- ポリシーの有効期限文字列 (省略可能、デフォルトは 1 時間)。

インポート

```
require '../vendor/autoload.php';

use Aws\S3\PostObjectV4;
use Aws\S3\S3Client;
```

サンプルコード

```
require '../vendor/autoload.php';

use Aws\S3\PostObjectV4;
use Aws\S3\S3Client;

$client = new S3Client([
    'profile' => 'default',
    'region' => 'us-east-1',
]);
$bucket = 'doc-example-bucket10';
$starts_with = 'user/eric/';
$client->listBuckets();

// Set defaults for form input fields.
$formInputs = ['acl' => 'public-read'];

// Construct an array of conditions for policy.
```

```
$options = [
    ['acl' => 'public-read'],
    ['bucket' => $bucket],
    ['starts-with', '$key', $starts_with],
];

// Set an expiration time (optional).
$expires = '+2 hours';

$postObject = new PostObjectV4(
    $client,
    $bucket,
    $formInputs,
    $options,
    $expires
);

// Get attributes for the HTML form, for example, action, method, enctype.
$formAttributes = $postObject->getFormAttributes();

// Get attributes for the HTML form values.
$formInputs = $postObject->getFormInputs();
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>PHP</title>
</head>
<body>
<form action="<?php echo $formAttributes['action'] ?>" method="<?php echo
 $formAttributes['method'] ?>"
      enctype="<?php echo $formAttributes['enctype'] ?>">
    <label id="key">
        <input hidden type="text" name="key" value="<?php echo $starts_with ?><?php
 echo $formInputs['key'] ?>"/>
    </label>
    <h3>$formInputs:</h3>
    acl: <label id="acl">
        <input readonly type="text" name="acl" value="<?php echo $formInputs['acl'] ?
>"/>
    </label><br/>
    X-Amz-Credential: <label id="credential">
```

```
        <input readonly type="text" name="X-Amz-Credential" value="<?php echo
 $formInputs['X-Amz-Credential'] ?>"/>
    </label><br/>
    X-Amz-Algorithm: <label id="algorithm">
        <input readonly type="text" name="X-Amz-Algorithm" value="<?php echo
 $formInputs['X-Amz-Algorithm'] ?>"/>
    </label><br/>
    X-Amz-Date: <label id="date">
        <input readonly type="text" name="X-Amz-Date" value="<?php echo $formInputs['X-
Amz-Date'] ?>"/>
    </label><br/><br/><br/>
    Policy: <label id="policy">
        <input readonly type="text" name="Policy" value="<?php echo
 $formInputs['Policy'] ?>"/>
    </label><br/>
    X-Amz-Signature: <label id="signature">
        <input readonly type="text" name="X-Amz-Signature" value="<?php echo
 $formInputs['X-Amz-Signature'] ?>"/>
    </label><br/><br/>
    <h3>Choose file:</h3>
    <input type="file" name="file"/> <br/><br/>
    <h3>Upload file:</h3>
    <input type="submit" name="submit" value="Upload to Amazon S3"/>
</form>
</body>
</html>
```

AWS SDK for PHP バージョン 3 での静的ウェブホストとしての Amazon S3 バケットの使用

静的ウェブサイトを Amazon S3 上でホスティングすることができます。詳細については、「Amazon S3 での静的ウェブサイトのホスティング」を参照してください。

以下の例では、次の方法を示しています。

- を使用してバケットのウェブサイト設定を取得しますGetBucketWebsite。
- を使用してバケットのウェブサイト設定を設定しますPutBucketWebsite。
- を使用してバケットからウェブサイト設定を削除しますDeleteBucketWebsite。

AWS SDK for PHP バージョン 3 のすべてのサンプルコードは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します。AWS SDK for PHP バージョン 3 の認証情報を参照してください。

バケットのウェブサイト設定の取得、設定および削除

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\S3\S3Client;
```

サンプルコード

```
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

// Retrieving the Bucket Website Configuration
$bucket = 'my-s3-bucket';
try {
    $resp = $s3Client->getBucketWebsite([
        'Bucket' => $bucket
    ]);
    echo "Succeed in retrieving website configuration for bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

// Setting a Bucket Website Configuration
$params = [
    'Bucket' => $bucket,
    'WebsiteConfiguration' => [
        'ErrorDocument' => [
            'Key' => 'foo',
```

```
        ],
        'IndexDocument' => [
            'Suffix' => 'bar',
        ],
    ]
];

try {
    $resp = $s3Client->putBucketWebsite($params);
    echo "Succeed in setting bucket website configuration.\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Deleting a Bucket Website Configuration
try {
    $resp = $s3Client->deleteBucketWebsite([
        'Bucket' => $bucket
    ]);
    echo "Succeed in deleting policy for bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS SDK for PHP バージョン 3 での Amazon S3 バケットポリシーの使用

バケットポリシーを使用して、Amazon S3 リソースに対するアクセス許可を付与できます。詳細については、「バケットポリシーとユーザーポリシーの使用」を参照してください。

以下の例では、次の方法を示しています。

- を使用して、指定されたバケットのポリシーを返しますGetBucketPolicy。
- を使用してバケットのポリシーを置き換えますPutBucketPolicy。
- を使用してバケットからポリシーを削除しますDeleteBucketPolicy。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

バケットのポリシーの取得、削除、および置き換え

インポート

```
require "vendor/autoload.php";

use Aws\Exception\AwsException;
use Aws\S3\S3Client;
```

サンプルコード

```
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

$bucket = 'my-s3-bucket';

// Get the policy of a specific bucket
try {
    $resp = $s3Client->getBucketPolicy([
        'Bucket' => $bucket
    ]);
    echo "Succeed in receiving bucket policy:\n";
    echo $resp->get('Policy');
    echo "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Deletes the policy from the bucket
try {
    $resp = $s3Client->deleteBucketPolicy([
```

```
        'Bucket' => $bucket
    ]);
    echo "Succeed in deleting policy of bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Replaces a policy on the bucket
try {
    $resp = $s3Client->putBucketPolicy([
        'Bucket' => $bucket,
        'Policy' => 'foo policy',
    ]);
    echo "Succeed in put a policy on bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}
```

AWS SDK for PHP バージョン 3 での S3 アクセスポイント ARN の使用

S3 では、S3 バケットと対話する新しい方法であるアクセスポイントが導入されました。アクセスポイントには、バケットに直接ではなく、一意のポリシーと設定を適用できます。AWS SDK for PHP では、バケット名を明示的に指定するのではなく、バケットフィールドでアクセスポイントの ARN を API オペレーションに使用できます。S3 アクセスポイントと ARN の仕組みの詳細については、こちらを参照してください 以下の例では、次の方法を示しています。

- アクセスポイント ARN GetObjectで を使用して、バケットからオブジェクトを取得します。
- アクセスポイント ARN PutObjectで を使用して、バケットにオブジェクトを追加します。
- クライアントリージョンの代わりに ARN リージョンを使用するように S3 クライアントを設定します。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

インポート

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
```

Get Object

まず AWS リージョンとバージョンを指定する AWS.S3 クライアントサービスを作成します。次に、キーと Bucket フィールドの S3 アクセスポイントの ARN を指定して getObject メソッドを呼び出します。これにより、そのアクセスポイントに関連付けられたバケットからオブジェクトが取得されます。

サンプルコード

```
$s3 = new S3Client([
    'version'     => 'latest',
    'region'      => 'us-west-2',
]);
$result = $s3->getObject([
    'Bucket' => 'arn:aws:s3:us-west-2:123456789012:accesspoint:endpoint-name',
    'Key' => 'MyKey'
]);
```

バケットにオブジェクトを配置する

まず AWS リージョンとバージョンを指定する AWS.S3 クライアントサービスを作成します。次に、必要なキー、本文やソースファイル、および Bucket フィールドに S3 アクセスポイントの ARN を指定して putObject メソッドを呼び出します。これにより、そのアクセスポイントに関連付けられたバケットにオブジェクトが格納されます。

サンプルコード

```
$s3 = new S3Client([
```

```
    'version'     => 'latest',
    'region'      => 'us-west-2',
]);
$result = $s3->putObject([
    'Bucket' => 'arn:aws:s3:us-west-2:123456789012:accesspoint:endpoint-name',
    'Key' => 'MyKey',
    'Body' => 'MyBody'
]);
```

クライアントリージョンの代わりに ARN リージョンを使用するように S3 クライアントを設定する

S3 クライアントオペレーションで S3 アクセスポイント の ARN を使用する場合、デフォルトでは、クライアントは ARN リージョンがクライアントリージョンと一致していることを確認し、一致していない場合は例外をスローします。この動作は、use_arn_region 設定オプションを true に設定することで、クライアントリージョン上で ARN リージョンを受け入れるように変更できます。デフォルトでは、オプションは false に設定されています。

サンプルコード

```
$s3 = new S3Client([
    'version'        => 'latest',
    'region'         => 'us-west-2',
    'use_arn_region' => true
]);
```

クライアントは、環境変数と Config ファイルオプションも次の優先順位でチェックします。

1. 上記の例のような、use_arn_region クライアントオプション。
2. 環境変数 AWS_S3_USE_ARN_REGION

```
export AWS_S3_USE_ARN_REGION=true
```

1. s3_use_arn_region 共有設定ファイルの config 変数 AWS (デフォルトでは ~/.aws/config にあります)。

```
[default]
s3_use_arn_region = true
```

AWS SDK for PHP バージョン 3 で Amazon S3 マルチリージョンアクセスポイントを使用する

Amazon Simple Storage Service (S3) マルチリージョンアクセスポイントは、 間で Amazon S3 リクエストトラフィックをルーティングするためのグローバルエンドポイントを提供しますAWS リージョン。

マルチリージョンアクセスポイントは、 SDK for PHP、別の SDK、S3 コンソール、または AWS CLI を使用して作成できます。 AWS

> **Important**
>
> SDK for PHP でマルチリージョンアクセスポイントを使用するには、PHP 環境に AWS Common Runtime (AWS CRT) 拡張機能がインストールされている必要があります。

マルチリージョンアクセスポイントを作成すると、Amazon S3 は次の形式の Amazon リソースネーム (ARN) を生成します。

`arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`

生成された ARN は、 getObject()および putObject()メソッドのバケット名の代わりに使用できます。

```
<?php
require './vendor/autoload.php';

use Aws\S3\S3Client;

// Assign the Multi-Region Access Point to a variable and use it place of a bucket
 name.
$mrap = 'arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap';
$key = 'my-key';

$s3Client = new S3Client([
    'region' => 'us-east-1'
]);

$s3Client->putObject([
    'Bucket' => $mrap,
    'Key' => $key,
    'Body' => 'Hello World!'
```

```
]);

$result = $s3Client->getObject([
    'Bucket' => $mrap,
    'Key' => $key
]);

echo $result['Body'] . "\n";

// Clean up.
$result = $s3Client->deleteObject([
    'Bucket' => $mrap,
    'Key' => $key
]);

$s3Client->waitUntil('ObjectNotExists', ['Bucket' => $mrap, 'Key' => $key]);

echo "Object deleted\n";
```

Secrets Manager API および AWS SDK for PHP バージョン 3 を使用したシークレットの管理

AWS Secrets Manager は、パスワード、API キー、データベース認証情報などの共有されたシークレットを保存し、管理します。Secrets Manager サービスにより、開発者はデプロイされたコードのハードコードされた認証情報を、Secrets Manager への埋め込みの呼び出しで置き換えることができます。

Secrets Manager では、Amazon Relational Database Service (Amazon RDS) データベースの認証情報のスケジュールされた自動ローテーションがネイティブにサポートされ、アプリケーションのセキュリティが向上します。また、Secrets Manager は、AWS Lambda を使って他のデータベースやサードパーティーのサービスのシークレットをシームレスにローテーションさせ、サービス固有の詳細を実装することができます。

以下の例では、次の方法を示しています。

- CreateSecret を使用してシークレットを作成します。
- GetSecretValue を使用してシークレットを取得します。
- ListSecrets を使用して、Secrets Manager によって保存されたすべてのシークレットをリストします。
- DescribeSecret を使用して、指定されたシークレットの詳細を取得します。

- PutSecretValue を使用して、指定されたシークレットを更新します。
- RotateSecret を使用してシークレットの更新を設定します。
- DeleteSecret を使用して、シークレットを削除対象としてマークします。

AWS SDK for PHP 用のすべてのサンプルコードは GitHub で入手できます。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Secrets Manager でのシークレットの作成

Secrets Manager でシークレットを作成するには、CreateSecret オペレーションを使用します。

この例では、ユーザー名とパスワードは JSON 文字列として保存されます。

インポート

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);
$secretName = 'MySecretName';
$secret = json_encode([
    "username" => getenv("SMDEMO_USERNAME"),
    "password" => getenv("SMDEMO_PASSWORD"),
]);
$description = '<<Description>>';
try {
    $result = $client->createSecret([
        'Description' => $description,
        'Name' => $secretName,
        'SecretString' => $secret,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Secrets Manager からのシークレットの取得

Secrets Manager に保存されているシークレットの値を取得するには、GetSecretValue オペレーションを使用します。

次の例では、secret シークレットは保存された値を含む文字列です。username の値が <<USERNAME>> で、password の値が <<PASSWORD>> の場合、secret 出力は 次のようになります。

```
{"username":"<<USERNAME>>","password":"<<PASSWORD>>"}
```

json_decode($secret, true) を使用して配列の値にアクセスします。

インポート

```
require 'vendor/autoload.php';

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-east-1',
]);

$secretName = 'MySecretName';

try {
    $result = $client->getSecretValue([
        'SecretId' => $secretName,
```

```
    ]);
} catch (AwsException $e) {
    $error = $e->getAwsErrorCode();
    if ($error == 'DecryptionFailureException') {
        // Secrets Manager can't decrypt the protected secret text using the provided
 AWS KMS key.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'InternalServiceErrorException') {
        // An error occurred on the server side.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'InvalidParameterException') {
        // You provided an invalid value for a parameter.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'InvalidRequestException') {
        // You provided a parameter value that is not valid for the current state of
 the resource.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'ResourceNotFoundException') {
        // We can't find the resource that you asked for.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
}
// Decrypts secret using the associated KMS CMK.
// Depending on whether the secret is a string or binary, one of these fields will be
 populated.
if (isset($result['SecretString'])) {
    $secret = $result['SecretString'];
} else {
    $secret = base64_decode($result['SecretBinary']);
}
print $secret;
$secretArray = json_decode($secret, true);
$username = $secretArray['username'];
$password = $secretArray['password'];
```

```
// Your code goes here;
```

Secrets Manager に保存されているシークレットのリスト

Secrets Manager によって保存されたすべてのシークレットのリストを取得するには、ListSecrets オペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);

try {
    $result = $client->listSecrets([
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

シークレットに関する詳細情報の取得

保存されたシークレットには、更新ルールに関するメタデータ、最終アクセス時間または変更時間、ユーザーが作成したタグ、および Amazon リソースネーム (ARN) が含まれています。Secrets Manager に保存されている、指定されたシークレットの詳細を取得するには、DescribeSecret オペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);

$secretName = 'MySecretName';

try {
    $result = $client->describeSecret([
        'SecretId' => $secretName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

シークレット値を更新する

新しく暗号化されたシークレットの値を Secrets Manager に保存するには、PutSecretValue オペレーションを使用します。

これにより、シークレットの新しいバージョンが作成されます。シークレットのバージョンが既に存在する場合、`VersionStages` パラメータと `AWSCURRENT` の値を追加し、この値を取得するときに新しい値が使用されるようにします。

インポート

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);
$secretName = 'MySecretName';
$secret = json_encode([
    "username" => getenv("SMDEMO_USERNAME"),
    "password" => getenv("SMDEMO_PASSWORD"),
]);
try {
    $result = $client->putSecretValue([
        'SecretId' => $secretName,
        'SecretString' => $secret,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Secrets Manager の既存のシークレット値のローテーション

Secrets Manager に保存されている既存のシークレットの値をローテーションするには、Lambda ローテーション関数と RotateSecret オペレーションを使用します。

開始する前に、シークレットをローテーションする Lambda 関数を作成します。現在、AWS コードサンプルカタログには、Amazon RDS データベースの認証情報をローテーションするための複数の Lambda コード例が含まれています。

> **Note**
>
> シークレットのローテーションの詳細については、「AWS Secrets Manager ユーザーガイド」の「AWS Secrets Manager シークレットのローテーション」を参照してください。

Lambda 関数を設定したら、新しいシークレットのローテーションを設定します。

インポート

```
require 'vendor/autoload.php';

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);

$secretName = 'MySecretName';
$lambda_ARN = 'arn:aws:lambda:us-
west-2:123456789012:function:MyTestDatabaseRotationLambda';
$rules = ['AutomaticallyAfterDays' => 30];

try {
    $result = $client->rotateSecret([
        'RotationLambdaARN' => $lambda_ARN,
        'RotationRules' => $rules,
        'SecretId' => $secretName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

更新が設定されている場合、RotateSecret オペレーションを使用して更新を実装できます。

インポート

```
require 'vendor/autoload.php';

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);

$secretName = 'MySecretName';

try {
    $result = $client->rotateSecret([
        'SecretId' => $secretName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Secrets Manager からのシークレットの削除

指定されたシークレットを Secrets Manager から削除するには、DeleteSecret オペレーションを使用します。シークレットが誤って削除されないようにするため、削除を元に戻すことが可能な復旧期間を指定できる DeletionDate スタンプが、自動的にシークレットに追加されます。復旧期間を指定しない場合のデフォルトの期間は、30 日です。

インポート

```
require 'vendor/autoload.php';

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

サンプルコード

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
```

```
]);

$secretName = 'MySecretName';

try {
    $result = $client->deleteSecret([
        'SecretId' => $secretName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

関連情報

AWS SDK for PHP の例では、AWS Secrets Manager API リファレンスの以下の REST オペレーションを使用しています。

- CreateSecret
- GetSecretValue
- ListSecrets
- DescribeSecret
- PutSecretValue
- RotateSecret
- DeleteSecret

AWS Secrets Manager の使用の詳細については、AWS Secrets Manager ユーザーガイドを参照してください。

AWS SDK for PHP バージョン 3 を使用した Amazon SES のコードサンプル

Amazon Simple Email Service (Amazon SES) は、ユーザー自身の E メールアドレスとドメインを使用して E メールを送受信するための、簡単で費用を削減できる方法を提供する E メールプラットフォームです。Amazon SES の詳細については、「Amazon SES デベロッパーガイド」を参照してください。

AWS には Amazon SES サービスの 2 つのバージョンがあり、これに対応して SDK for PHP には SESClient と SESv2Client という 2 つのバージョンのクライアントが用意されています。メソッドの呼び出し方法や結果は異なる場合がありますが、多くの場合、クライアントの機能は重複しています。この 2 つの API は独自の機能も備えているため、両方のクライアントを使用してすべての機能にアクセスできます。

このセクションの例ではすべてオリジナルの `SesClient` を使用しています。

AWS SDK for PHP バージョン 3 用のすべてのサンプルコードは GitHub で入手できます。

トピック

- Amazon SES API および AWS SDK for PHP バージョン 3 を使用した E メールアイデンティティの検証
- Amazon SES API と AWS SDK for PHP バージョン 3 を使用したカスタムメールテンプレートの作成
- Amazon SES API および AWS SDK for PHP バージョン 3 を使用した E メールフィルターの管理
- Amazon SES API および AWS SDK for PHP バージョン 3 を使用した E メールルールの作成と管理
- Amazon SES API と AWS SDK for PHP バージョン 3 を使用した送信活動のモニタリング
- Amazon SES API および AWS SDK for PHP バージョン 3 を使用した送信者の認証

Amazon SES API および AWS SDK for PHP バージョン 3 を使用した E メールアイデンティティの検証

初めて Amazon Simple Email Service (Amazon SES) アカウントを使用し始めるとき、E メールの送信先となる同じ AWS リージョン内のすべての送信者と受取人を確認する必要があります。E メール送信の詳細については、「Amazon SES を使用してメールを送信する」を参照してください。

以下の例では、次の方法を示しています。

- を使用して E メールアドレスを検証しますVerifyEmailIdentity。
- を使用して E メールドメインを検証しますVerifyDomainIdentity。
- を使用してすべての E メールアドレスを一覧表示しますListIdentities。
- を使用してすべての E メールドメインを一覧表示しますListIdentities。
- を使用して E メールアドレスを削除しますDeleteIdentity。
- を使用して E メールドメインを削除しますDeleteIdentity。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Amazon SES の使用の詳細については、「Amazon SES デベロッパーガイド」を参照してください。

E メールアドレスを検証する

Amazon SES は、確認済み E メールアドレスまたはドメインからのみ E メールを送信できます。E メールアドレスを確認することで、そのアドレスの所有者であることを示し、そのアドレスから E メールを送信することを Amazon SES に許可します。

以下のコード例を実行すると、指定したアドレスに Amazon SES により E メールが送信されます。お客様 (または E メールの受取人) が E メール内のリンクをクリックすると、アドレスが確認されます。

Amazon SES アカウントに E メールアドレスを追加するには、 VerifyEmailIdentityオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$email = 'email_address';
```

```
try {
    $result = $SesClient->verifyEmailIdentity([
        'EmailAddress' => $email,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

E メールドメインを検証する

Amazon SES は、確認済み E メールアドレスまたはドメインからのみ E メールを送信できます。ドメインを確認することで、そのドメインの所有者であることを示します。ドメインを検証すると、そのドメインの任意のアドレスからの E メールの送信を Amazon SES に許可します。

以下のコード例を実行すると、Amazon SES により検証トークンが提供されます。ドメインの DNS 設定にトークンを追加する必要があります。詳細については、「Amazon Simple Email Service デベロッパーガイド」の「Amazon SES でのドメインの検証」を参照してください。

Amazon SES アカウントに送信ドメインを追加するには、 VerifyDomainIdentityオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);
```

```
$domain = 'domain.name';

try {
    $result = $SesClient->verifyDomainIdentity([
        'Domain' => $domain,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

E メールアドレスをリストする

検証ステータスに関係なく、現在のAWSリージョンで送信された E メールアドレスのリストを取得するには、 ListIdentitiesオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

try {
    $result = $SesClient->listIdentities([
        'IdentityType' => 'EmailAddress',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    echo $e->getMessage();
    echo "\n";
}
```

E メールドメインをリストする

検証ステータスに関係なく、現在のAWSリージョンで送信された E メールドメインのリストを取得するには、 ListIdentitiesオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

try {
    $result = $SesClient->listIdentities([
        'IdentityType' => 'Domain',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

E メールアドレスを削除する

ID のリストから検証済み E メールアドレスを削除するには、 DeleteIdentityオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$email = 'email_address';

try {
    $result = $SesClient->deleteIdentity([
        'Identity' => $email,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

E メールドメインを削除する

検証済み ID のリストから検証済み E メールドメインを削除するには、 DeleteIdentityオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$domain = 'domain.name';

try {
    $result = $SesClient->deleteIdentity([
        'Identity' => $domain,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Amazon SES API と AWS SDK for PHP バージョン 3 を使用したカスタムメールテンプレートの作成

Amazon Simple Email Service (Amazon SES) では、テンプレートを使用して受取人ごとにパーソナライズされた E メールを送信できます。テンプレートには、件名、E メール本文のテキストパートと HTML パートが含まれています。件名および本文セクションには、受取人ごとにパーソナライズされたユニークな値を含めることもできます。

詳細については、「Amazon Simple Email Service デベロッパーガイド」の「Amazon SES を使用してパーソナライズされた E メールを送信する」を参照してください。

以下の例では、次の方法を示しています。

- を使用して E メールテンプレートを作成しますCreateTemplate。
- を使用してすべての E メールテンプレートを一覧表示しますListTemplates。
- を使用して E メールテンプレートを取得しますGetTemplate。
- を使用して E メールテンプレートを更新しますUpdateTemplate。
- を使用して E メールテンプレートを削除しますDeleteTemplate。

- を使用してテンプレート化された E メールを送信しますSendTemplatedEmail。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Amazon SES の使用の詳細については、「Amazon SES デベロッパーガイド」を参照してください。

E メールテンプレートを作成する

パーソナライズされた E メールメッセージを送信するテンプレートを作成するには、CreateTemplateオペレーションを使用します。このテンプレートは、テンプレートが追加された AWS リージョンでのメッセージの送信が承認されたどのアカウントでも使用できます。

> **Note**
>
> Amazon SES は HTML を検証しないため、E メールを送信する前に HtmlPart が有効であることを確認してください。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);
```

```
$name = 'Template_Name';
$html_body = '<h1>AWS Amazon Simple Email Service Test Email</h1>' .
    '<p>This email was sent with <a href="https://aws.amazon.com/ses/">' .
    'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-php/">' .
    'AWS SDK for PHP</a>.</p>';
$subject = 'Amazon SES test (AWS SDK for PHP)';
$plaintext_body = 'This email was send with Amazon SES using the AWS SDK for PHP.';

try {
    $result = $SesClient->createTemplate([
        'Template' => [
            'HtmlPart' => $html_body,
            'SubjectPart' => $subject,
            'TemplateName' => $name,
            'TextPart' => $plaintext_body,
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

E メールテンプレートを取得する

件名、HTML 本文、プレーンテキストを含む既存の E メールテンプレートの内容を表示するには、GetTemplateオペレーションを使用します。のみ必須 TemplateName です。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
```

```
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Template_Name';

try {
    $result = $SesClient->getTemplate([
        'TemplateName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

すべての E メールテンプレートをリストする

現在のAWSリージョンAWS アカウントで に関連付けられているすべての E メールテンプレートのリストを取得するには、 ListTemplatesオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

try {
    $result = $SesClient->listTemplates([
        'MaxItems' => 10,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

E メールテンプレートを更新する

件名、HTML 本文、プレーンテキストなど、特定の E メールテンプレートの内容を変更するには、UpdateTemplateオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Template_Name';
$html_body = '<h1>AWS Amazon Simple Email Service Test Email</h1>' .
    '<p>This email was sent with <a href="https://aws.amazon.com/ses/">' .
    'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-php/">' .
    'AWS SDK for PHP</a>.</p>';
$subject = 'Amazon SES test (AWS SDK for PHP)';
$plaintext_body = 'This email was send with Amazon SES using the AWS SDK for PHP.';

try {
    $result = $SesClient->updateTemplate([
        'Template' => [
            'HtmlPart' => $html_body,
            'SubjectPart' => $subject,
```

```
            'TemplateName' => $name,
            'TextPart' => $plaintext_body,
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

E メールテンプレートを削除する

特定の E メールテンプレートを削除するには、 DeleteTemplateオペレーションを使用します。必要なのは だけです TemplateName。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Template_Name';

try {
    $result = $SesClient->deleteTemplate([
        'TemplateName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
```

```
    echo "\n";
}
```

テンプレートを使用して E メールを送信する

テンプレートを使用して受信者に E メールを送信するには、 SendTemplatedEmailオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$template_name = 'Template_Name';
$sender_email = 'email_address';
$recipient_emails = ['email_address'];

try {
    $result = $SesClient->sendTemplatedEmail([
        'Destination' => [
            'ToAddresses' => $recipient_emails,
        ],
        'ReplyToAddresses' => [$sender_email],
        'Source' => $sender_email,

        'Template' => $template_name,
        'TemplateData' => '{ }'
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    echo $e->getMessage();
    echo "\n";
}
```

Amazon SES API および AWS SDK for PHP バージョン 3 を使用した E メールフィルターの管理

E メールの送信に加えて、Amazon Simple Email Service (Amazon SES) を使用して E メールを受信することもできます。IP アドレスフィルタを使用すると、オプションで、特定の IP アドレスまたは IP アドレス範囲から送信されるメールを受け入れるか拒否するかを指定できます。詳細については、「Amazon SES による E メール受信の IP アドレスフィルターの管理」を参照してください。

以下の例では、次の方法を示しています。

- を使用して E メールフィルターを作成しますCreateReceiptFilter。
- を使用してすべての E メールフィルターを一覧表示しますListReceiptFilters。
- を使用して E メールフィルターを削除しますDeleteReceiptFilter。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Amazon SES の使用の詳細については、「Amazon SES デベロッパーガイド」を参照してください。

E メールフィルターを作成する

特定の IP アドレスからの E メールを許可またはブロックするには、 CreateReceiptFilterオペレーションを使用します。このフィルターを識別するための IP アドレスまたはアドレスの範囲と、一意の名前入力します。

インポート

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$filter_name = 'FilterName';
$ip_address_range = '10.0.0.1/24';

try {
    $result = $SesClient->createReceiptFilter([
        'Filter' => [
            'IpFilter' => [
                'Cidr' => $ip_address_range,
                'Policy' => 'Block|Allow',
            ],
            'Name' => $filter_name,
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

すべての E メールフィルターをリストする

現在のAWSリージョンAWS アカウントの に関連付けられている IP アドレスフィルターを一覧表示するには、 ListReceiptFiltersオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

try {
    $result = $SesClient->listReceiptFilters();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

E メールフィルターを削除する

特定の IP アドレスの既存のフィルターを削除するには、 DeleteReceiptFilterオペレーションを使用します。削除する受信フィルターを識別するための一意のフィルター名を入力します。

フィルタリングされるアドレス範囲を変更する必要がある場合、受信フィルターを削除して新たに作成することができます。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
```

```
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$filter_name = 'FilterName';

try {
    $result = $SesClient->deleteReceiptFilter([
        'FilterName' => $filter_name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Amazon SES API および AWS SDK for PHP バージョン 3 を使用した E メールルールの作成と管理

E メールの送信に加えて、Amazon Simple Email Service (Amazon SES) を使用して E メールを受信することもできます。受信ルールを使用すると、所有する E メールアドレスやドメインで受信した E メールを Amazon SES で処理する方法を指定できます。ルールは、他の AWS サービス (Amazon S3、Amazon SNS、AWS Lambda が含まれますが、これらに限定されません) に E メールを送信できます。

詳細については、「Amazon SES による E メール受信の受信ルールセットの管理」と「Amazon SES E メール受信のための受信ルールの管理」を参照してください。

以下の例では、次の方法を示しています。

- を使用して受信ルールセットを作成しますCreateReceiptRuleSet。
- を使用して受信ルールを作成しますCreateReceiptRule。
- を使用して受信ルールセットを記述しますDescribeReceiptRuleSet。
- を使用して受信ルールを記述しますDescribeReceiptRule。
- を使用して、すべての受信ルールセットを一覧表示しますListReceiptRuleSets。
- を使用して受信ルールを更新しますUpdateReceiptRule。
- を使用して受信ルールを削除しますDeleteReceiptRule。

- を使用して受信ルールセットを削除しますDeleteReceiptRuleSet。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Amazon SES の使用の詳細については、「Amazon SES デベロッパーガイド」を参照してください。

受信ルールセットを作成する

受信ルールセットには、受信ルールのコレクションが含まれています。受信ルールを作成するには、アカウントに少なくとも 1 つの受信ルールセットが関連付けられている必要があります。受信ルールセットを作成するには、一意の を指定 RuleSetName し、 CreateReceiptRuleSetオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Rule_Set_Name';

try {
    $result = $SesClient->createReceiptRuleSet([
        'RuleSetName' => $name,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

受信ルールの作成

受信ルールを既存の受信ルールセットに追加することで、受信 E メールを制御します。この例は、受信メッセージを Amazon S3 バケットに送信する受信ルールを作成する方法を示していますが、Amazon SNS と AWS Lambda にメッセージを送信することもできます。受信ルールを作成するには、 CreateReceiptRuleオペレーション RuleSetName にルールと を指定します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$rule_name = 'Rule_Name';
$rule_set_name = 'Rule_Set_Name';
$s3_bucket = 'Bucket_Name';

try {
    $result = $SesClient->createReceiptRule([
        'Rule' => [
            'Actions' => [
                [
                    'S3Action' => [
                        'BucketName' => $s3_bucket,
```

```
                    ],
                ],
            ],
            'Name' => $rule_name,
            'ScanEnabled' => true,
            'TlsPolicy' => 'Optional',
            'Recipients' => ['<string>']
        ],
        'RuleSetName' =>  $rule_set_name,

     ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

受信ルールセットを記述する

1 秒に 1 回、指定された受信ルールセットの詳細を返します。DescribeReceiptRuleSet オペレーションを使用するには、 を指定します RuleSetName。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Rule_Set_Name';

try {
```

```
    $result = $SesClient->describeReceiptRuleSet([
        'RuleSetName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

受信ルールを記述する

指定された受信ルールの詳細を返します。DescribeReceiptRule オペレーションを使用するには、RuleName と を指定します RuleSetName。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$rule_name = 'Rule_Name';
$rule_set_name = 'Rule_Set_Name';

try {
    $result = $SesClient->describeReceiptRule([
        'RuleName' => $rule_name,
        'RuleSetName' => $rule_set_name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    echo $e->getMessage();
    echo "\n";
}
```

すべての受信ルールセットをリストする

現在のAWSリージョンの に存在する受信ルールセットAWS アカウントを一覧表示するには、ListReceiptRuleSetsオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

try {
    $result = $SesClient->listReceiptRuleSets();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

受信ルールを更新する

この例は、受信メッセージを AWS Lambda 関数に送信する受信ルールを更新する方法を示していますが、Amazon SNS と Amazon S3 にメッセージを送信することもできます。UpdateReceiptRule オペレーションを使用するには、新しい受信ルールと を指定します RuleSetName。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$rule_name = 'Rule_Name';
$rule_set_name = 'Rule_Set_Name';
$lambda_arn = 'Amazon Resource Name (ARN) of the AWS Lambda function';
$sns_topic_arn = 'Amazon Resource Name (ARN) of the Amazon SNS topic';

try {
    $result = $SesClient->updateReceiptRule([
        'Rule' => [
            'Actions' => [
                'LambdaAction' => [
                    'FunctionArn' => $lambda_arn,
                    'TopicArn' => $sns_topic_arn,
                ],
            ],
            'Enabled' => true,
            'Name' => $rule_name,
            'ScanEnabled' => false,
            'TlsPolicy' => 'Require',
        ],
        'RuleSetName' => $rule_set_name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

受信ルールセットを削除する

現在無効になっていない指定された受信ルールセットを削除します。これにより、それに含まれるすべての受信ルールも削除されます。受信ルールセットを削除するには、 DeleteReceiptRuleSetオペレーション RuleSetName に を指定します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Rule_Set_Name';

try {
    $result = $SesClient->deleteReceiptRuleSet([
        'RuleSetName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

受信ルールを削除する

指定された受信ルールを削除するには、 DeleteReceiptRule オペレーション RuleSetName に RuleName と を指定します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

サンプルコード

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$rule_name = 'Rule_Name';
$rule_set_name = 'Rule_Set_Name';

try {
    $result = $SesClient->deleteReceiptRule([
        'RuleName' => $rule_name,
        'RuleSetName' => $rule_set_name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Amazon SES API と AWS SDK for PHP バージョン 3 を使用した送信活動のモニタリング

Amazon Simple Email Service (Amazon SES) は、送信アクティビティをモニタリングするための方法を提供しています。これらの方法を実装し、アカウントのバウンス率、苦情率、拒否率などの重要な指標を追跡することをお勧めします。バウンス率や苦情率が高すぎると、Amazon SES での E メール送信に支障が生じる場合があります。

以下の例では、次の方法を示しています。

- を使用して送信クォータを確認しますGetSendQuota。
- を使用して送信アクティビティをモニタリングしますGetSendStatistics。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Amazon SES の使用の詳細については、「Amazon SES デベロッパーガイド」を参照してください。

送信クォータを確認する

24 時間以内に送信できるメッセージの量には一定の制限があります。送信できるメッセージの数を確認するには、 GetSendQuotaオペレーションを使用します。詳細については、「Amazon SES の送信制限の管理」を参照してください。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

サンプルコード

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'

]);

try {
    $result = $SesClient->getSendQuota();
    $send_limit = $result["Max24HourSend"];
    $sent = $result["SentLast24Hours"];
```

```
    $available = $send_limit - $sent;
    print("<p>You can send " . $available . " more messages in the next 24 hours.</
p>");
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

送信アクティビティをモニタリングする

過去 2 週間に送信したメッセージのメトリクスを取得するには、 GetSendStatisticsオペレーションを使用します。この例では、配信試行数、バウンス数、苦情数、拒否されたメッセージ数が 15 分ごとに返されます。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

サンプルコード

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

try {
    $result = $SesClient->getSendStatistics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Amazon SES API および AWS SDK for PHP バージョン 3 を使用した送信者の認証

お客様の代わりに別の AWS アカウント、AWS Identity and Access Management ユーザー、または AWS サービスが Amazon Simple Email Service (Amazon SES) を介して E メールを送信できるようにするには、送信認可ポリシーを作成します。これは、お客様が所有しているアイデンティティにアタッチする JSON ドキュメントです。

このポリシーには、そのアイデンティティでの送信を許可するユーザーとその条件が明示的にリストされます。お客様とポリシーでアクセス許可を明示的に付与したエンティティ以外のすべての送信者は、E メールの送信が許可されません。アイデンティティにはポリシーをアタッチしないことも、1 つまたは複数のポリシーをアタッチすることもできます。さらに、複数のステートメントを含む 1 つのポリシーを作成して、複数のポリシーの効果を持たせることもできます。

詳細については、「Amazon SES での送信承認の使用」を参照してください。

以下の例では、次の方法を示しています。

- を使用して、承認された送信者を作成しますPutIdentityPolicy。
- を使用して、承認された送信者のポリシーを取得しますGetIdentityPolicies。
- を使用して、承認された送信者を一覧表示しますListIdentityPolicies。
- を使用して、承認された送信者のアクセス許可を取り消しますDeleteIdentityPolicy。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

Amazon SES の使用の詳細については、「Amazon SES デベロッパーガイド」を参照してください。

承認済み送信者を作成する

別の AWS アカウント がお客様の代わりに E メールを送信することを承認するには、アイデンティティポリシーを使用し、確認済みの E メールアドレスまたはドメインから E メールを送信する認可

を追加または更新します。ID ポリシーを作成するには、 PutIdentityPolicyオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

サンプルコード

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

$identity = "arn:aws:ses:us-east-1:123456789012:identity/example.com";
$other_aws_account = "0123456789";
$policy = <<<EOT
{
  "Id":"ExampleAuthorizationPolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AuthorizeAccount",
      "Effect":"Allow",
      "Resource":"$identity",
      "Principal":{
        "AWS":[ "$other_aws_account" ]
      },
      "Action":[
        "SES:SendEmail",
        "SES:SendRawEmail"
      ]
    }
  ]
}
EOT;
$name = "policyName";
```

```
try {
    $result = $SesClient->putIdentityPolicy([
        'Identity' => $identity,
        'Policy' => $policy,
        'PolicyName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

承認済み送信者のポリシーを取得する

特定の E メールアイデンティティまたはドメインアイデンティティに関連付けられている送信承認ポリシーを返します。特定の E メールアドレスまたはドメインの送信承認を取得するには、GetIdentityPolicyオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

サンプルコード

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

$identity = "arn:aws:ses:us-east-1:123456789012:identity/example.com";
$policies = ["policyName"];

try {
```

```
    $result = $SesClient->getIdentityPolicies([
        'Identity' => $identity,
        'PolicyNames' => $policies,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

承認済み送信者をリストする

現在のAWSリージョンの特定の E メール ID またはドメイン ID に関連付けられている送信承認ポリシーを一覧表示するには、 ListIdentityPoliciesオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

サンプルコード

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

$identity = "arn:aws:ses:us-east-1:123456789012:identity/example.com";

try {
    $result = $SesClient->listIdentityPolicies([
        'Identity' => $identity,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    echo $e->getMessage();
    echo "\n";
}
```

承認済み送信者のアクセス許可を取り消す

DeleteIdentityPolicy オペレーションに関連付けられた ID ポリシーを削除して、E メール ID またはドメイン ID を持つ E メールを送信AWS アカウントする別の の送信承認を削除します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

サンプルコード

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

$identity = "arn:aws:ses:us-east-1:123456789012:identity/example.com";
$name = "policyName";

try {
    $result = $SesClient->deleteIdentityPolicy([
        'Identity' => $identity,
        'PolicyName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS SDK for PHP バージョン 3 を使用した Amazon SNS のコードサンプル

Amazon Simple Notification Service (Amazon SNS) は、サブスクライブしているエンドポイントやクライアントへのメッセージ配信や送信を調整、管理するウェブサービスです。

Amazon SNS には、発行者 (生産者とも呼ばれます) とサブスクライバー (消費者とも呼ばれます) という 2 種類のクライアントが存在します。発行者は、論理アクセスポイントおよび通信チャネルであるトピックにメッセージを作成して送信することで、受信者と非同期的に通信します。サブスクライバー (ウェブサーバー、E メールアドレス、Amazon SQS キュー、AWS Lambda 関数) は、トピックにサブスクライブされているサポートされているプロトコル (Amazon SQS、HTTP/HTTPS URL、E メール、AWS SMS、Lambda) の 1 つを使用して、メッセージや通知を消費または受信します。

AWS SDK for PHP バージョン 3 用のすべてのサンプルコードは GitHub で入手できます。

トピック

- AWS SDK for PHP バージョン 3 を使用した Amazon SNS でのトピックの管理
- AWS SDK for PHP バージョン 3 を使用した Amazon SNS でのサブスクリプションの管理
- AWS SDK for PHP バージョン 3 による Amazon SNS での SMS メッセージの送信

AWS SDK for PHP バージョン 3 を使用した Amazon SNS でのトピックの管理

Amazon Simple Queue Service (Amazon SQS)、HTTP/HTTPS URL、E メール、AWS SMS または AWS Lambda に通知を送信するには、まずそのトピックのサブスクライバへのメッセージ配信を管理するトピックを管理する必要があります。

オブザーバー設計パターンの観点で言うと、トピックは件名と同様です。トピックが作成されたら、メッセージがトピックに発行されたときに自動的に通知を受け取るサブスクライバを追加します。

トピックへのサブスクライブの詳細については、「AWS SDK for PHP バージョン 3 を使用した Amazon SNS でのサブスクリプションの管理」を参照してください。

以下の例では、次の方法を示しています。

- を使用して に通知を発行するトピックを作成しますCreateTopic。
- を使用して、リクエスタのトピックのリストを返しますListTopics。
- を使用して、トピックとそのすべてのサブスクリプションを削除しますDeleteTopic。

- を使用して、トピックのすべてのプロパティを返しますGetTopicAttributes。
- トピック所有者が を使用してトピックの属性を新しい値に設定できるようにしますSetTopicAttributes。

Amazon SNS の使用の詳細については、「メッセージの配信ステータスの Amazon SNS トピック属性を使用するを」参照してください。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

トピックの作成

トピックを作成するには、 CreateTopicオペレーションを使用します。

AWS アカウント 内の各トピック名は一意にする必要があります。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

トピックをリストする

現在のAWSリージョンで最大 100 個の既存のトピックを一覧表示するには、 ListTopicsオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

トピックの削除

既存のトピックとそのすべてのサブスクリプションを削除するには、 DeleteTopicオペレーションを使用します。

サブスクライバにまだ配信されていないメッセージもすべて削除されます。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

トピック属性を取得する

単一の既存のトピックのプロパティを取得するには、 GetTopicAttributesオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

トピック属性を設定する

単一の既存のトピックのプロパティを更新するには、 SetTopicAttributesオペレーションを使用します。

Policy、DisplayName、および DeliveryPolicy 属性のみ設定できます。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
```

```
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 を使用した Amazon SNS でのサブスクリプションの管理

Amazon Simple Notification Service (Amazon SNS) トピックを使用して、Amazon Simple Queue Service (Amazon SQS)、HTTP/HTTPS、E メールアドレス、AWS Server Migration Service (AWS SMS)、または AWS Lambda に通知を送信します。

サブスクリプションは、サブスクライバへのメッセージの送信を管理するトピックにアタッチされます。トピックの作成の詳細については、「AWS SDK for PHP バージョン 3 を使用した Amazon SNS でのトピックの管理」を参照してください。

以下の例では、次の方法を示しています。

- Subscribe を使用して既存のトピックにサブスクライブする。
- を使用してサブスクリプションを検証しますConfirmSubscription。
- を使用して既存のサブスクリプションを一覧表示しますListSubscriptionsByTopic。
- Unsubscribe を使用してサブスクリプションを削除する。
- Publish を使用してトピックのすべてのサブスクライバにメッセージを送信する。

Amazon SNS の使用の詳細については、「Amazon SNS を使用したシステム間メッセージング」を参照してください。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

E メールアドレスを SNS トピックにサブスクライブする

E メールアドレスへのサブスクリプションを開始するには、Subscribe オペレーションを使用します。

subscribe メソッドを使用し、渡されるパラメータに使用する値に応じて Amazon SNS トピックに複数のさまざまなエンドポイントをサブスクライブできます。これは、このトピックの他の例に示されます。

この例では、エンドポイントは E メールアドレスです。確認トークンがこの E メールに送信されます。受け取ってから 3 日以内に、この確認トークンを使用してサブスクリプションを確認します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
```

```
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

アプリケーションエンドポイントをトピックにサブスクライブする

ウェブアプリへのサブスクリプションを開始するには、Subscribe オペレーションを使用します。

subscribe メソッドを使用し、渡されるパラメータに使用する値に応じて Amazon SNS トピックに複数のさまざまなエンドポイントをサブスクライブできます。これは、このトピックの他の例に示されます。

この例では、エンドポイントは URL です。確認トークンがこのウェブアドレスに送信されます。受け取ってから 3 日以内に、この確認トークンを使用してサブスクリプションを確認します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';
```

```
try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Lambda 関数をトピックにサブスクライブする

Lambda 関数へのサブスクリプションを開始するには、Subscribe オペレーションを使用します。

subscribe メソッドを使用し、渡されるパラメータに使用する値に応じて Amazon SNS トピックに複数のさまざまなエンドポイントをサブスクライブできます。これは、このトピックの他の例に示されます。

この例では、エンドポイントは Lambda 関数です。確認トークンがこの Lambda 関数に送信されます。受け取ってから 3 日以内に、この確認トークンを使用してサブスクリプションを確認します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
$protocol = 'lambda';
$endpoint = 'arn:aws:lambda:us-east-1:123456789023:function:messageStore';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

トピックにテキスト SMS をサブスクライブする

複数の電話番号に SMS メッセージを同時に送信するには、各番号をトピックにサブスクライブします。

電話番号へのサブスクリプションを開始するには、Subscribe オペレーションを使用します。

subscribe メソッドを使用し、渡されるパラメータに使用する値に応じて Amazon SNS トピックに複数のさまざまなエンドポイントをサブスクライブできます。これは、このトピックの他の例に示されます。

この例では、エンドポイントは E.164 形式 (国際的な音声通信の規格) の電話番号です。

確認トークンがこの電話番号に送信されます。受け取ってから 3 日以内に、この確認トークンを使用してサブスクリプションを確認します。

Amazon SNS を使用して SMS メッセージを送信するもう 1 つの方法については、「AWS SDK for PHP バージョン 3 を使用した Amazon SNS での SMS メッセージの送信」を参照してください。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'sms';
$endpoint = '+1XXX5550100';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

トピックへのサブスクリプションを確認する

実際にサブスクリプションを作成するには、エンドポイントの所有者が、前述のとおりサブスクリプションが最初に確立されたときに送信されるトークンを使用して、トピックからメッセージを受信する意思があることを確認する必要があります。確認トークンの有効期間は 3 日間です。3 日間後、新しいサブスクリプションを作成することでトークンを再送信することができます。

サブスクリプションを確認するには、 ConfirmSubscriptionオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

トピックへのサブスクリプションをリストする

特定のAWSリージョンで最大 100 の既存のサブスクリプションを一覧表示するには、ListSubscriptionsオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

トピックからサブスクリプションを解除する

トピックにサブスクライブされたエンドポイントを削除するには、Unsubscribe オペレーションを使用します。

サブスクリプションで削除のために認証が必要である場合、サブスクリプション解除できるのはサブスクリプションの所有者またはトピックの所有者のみであり、AWS 署名が必要です。サブスクリプション解除呼び出しに認証が必要なく、リクエスタがサブスクリプション所有者でない場合は、最終的なキャンセル確認メッセージがエンドポイントに配信されます。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Amazon SNS トピックへのメッセージの発行

Amazon SNS トピックにサブスクライブされた各エンドポイントにメッセージを配信するには、Publish オペレーションを使用します。

Amazon SNS トピックのメッセージテキストと Amazon リソースネーム (ARN) など、メッセージを発行するためのパラメータを含むオブジェクトを作成します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
```

```
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 による Amazon SNS での SMS メッセージの送信

Amazon Simple Notification Service (Amazon SNS) を使用して、SMS 対応デバイスにテキストメッセージ (SMS メッセージ) を送信できます。電話番号をトピックにサブスクライブし、トピックへメッセージを送信することにより、電話番号へメッセージを直接送信または、一度に複数の電話番号にメッセージを送信できます。

Amazon SNS を使用して、配信の最適化の方法 (コストに対してか、確実な配信に対してか)、毎月の使用量の上限、メッセージ配信がログに記録される方法、SMS の毎日の使用状況レポートをサブスクライブするかどうかなど、SMS メッセージのプリファレンスを指定します。これらのプリファレンスが取得され、Amazon SNS の SMS 属性として設定されます。

SMS メッセージを送信するときは、E.164 形式を使用して電話番号を指定します。E.164 は、国際的な音声通信に使用される電話番号の構造の規格です。この形式に従う電話番号には最大 15 桁を設定でき、プラス記号 (+) および国コードのプレフィックスがついています。たとえば、E.164 形式の米国の電話番号は +1001XXX5550100 として表示されます。

以下の例では、次の方法を示しています。

- GetSMSAttributes を使用して、アカウントから SMS メッセージを送信するためのデフォルト設定を取得する。
- SetSMSAttributes を使用して、アカウントから SMS メッセージを送信するためのデフォルト設定を更新する。
- 特定の電話番号の所有者が、CheckIfPhoneNumberISOptedOut を使用してアカウントからの SMS メッセージの受信をオプトアウトしたかどうかを確認します。
- 所有者が を使用してアカウントからの SMS メッセージの受信をオプトアウトした電話番号を一覧表示しますListPhoneNumberOptedOut。
- Publish を使用して、電話番号に直接テキストメッセージ (SMS) を送信する。

Amazon SNS の使用の詳細については、「受信者が携帯電話番号の場合のユーザー通知に Amazon SNS を使用する (SMS 送信)」を参照してください。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

SMS 属性の取得

SMS メッセージのデフォルト設定を取得するには、GetSMSAttributes オペレーションを使用します。

この例では、DefaultSMSType 属性を取得します。この属性は、SMS メッセージが Promotional (コストが最も低くなるようにメッセージ配信が最適化されます) として送信されるのか、Transactional (信頼性が最も高くなるようにメッセージ配信が最適化されます) として送信されるのかを制御します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

SMS 属性の設定

SMS メッセージのデフォルト設定を更新するには、SetSMSAttributes オペレーションを使用します。

この例では、DefaultSMSType 属性を Transactional に設定します。これにより、信頼性が最も高くなるようにメッセージ配信が最適化されます。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

電話番号がオプトアウトしているかどうかを確認する

特定の電話番号の所有者がアカウントからの SMS メッセージの受信をオプトアウトしたかどうかを判断するには、 CheckIfPhoneNumberIsOptedOutオペレーションを使用します。

この例では、電話番号は E.164 形式 (国際的な音声通信の規格) です。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

オプトアウトした電話番号を一覧表示する

所有者がアカウントからの SMS メッセージの受信をオプトアウトした電話番号のリストを取得するには、 ListPhoneNumbersOptedOutオペレーションを使用します。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

テキストメッセージ (SMS メッセージ) に発行する

電話番号に直接テキストメッセージ (SMS メッセージ) を配信するには、Publish オペレーションを使用します。

この例では、電話番号は E.164 形式 (国際的な音声通信の規格) です。

SMS メッセージには最大 140 バイト含めることができます。1 回の SMS 発行アクションのサイズ制限は、1,600 バイトです。

SMS メッセージの送信の詳細については、「SMS メッセージの送信」を参照してください。

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

サンプルコード

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 を使用した Amazon SQS のコード例

Amazon Simple Queue Service (SQS) は、高速で、信頼性が高く、スケーラブルな、フルマネージド型のメッセージキューイングサービスです。Amazon SQSを使用すると、クラウドアプリケーションのコンポーネントを分離できます。Amazon SQS には、高スループットおよび少なくとも 1 回処理の標準キュー、および FIFO (先入先出) 配信および正確に 1 回のみ処理の FIFO キューが含まれています。

AWS SDK for PHP バージョン 3 用のすべてのサンプルコードは GitHub で入手できます。

トピック

- AWS SDK for PHP バージョン 3 を使用した Amazon SQS でのロングポーリングの有効化
- AWS SDK for PHP バージョン 3 による Amazon SQS での可視性タイムアウトの管理
- AWS SDK for PHP バージョン 3 を使用した Amazon SQS でのメッセージの送受信
- AWS SDK for PHP バージョン 3 による Amazon SQS でのデッドレターキューの使用

- AWS SDK for PHP バージョン 3 による Amazon SQS でのキューの使用

AWS SDK for PHP バージョン 3 を使用した Amazon SQS でのロングポーリングの有効化

ロングポーリングは、レスポンスの送信前にメッセージがキューで使用可能になるまで Amazon SQS が指定された時間待機できるようにすることで、空のレスポンス数を削減します。また、ロングポーリングでは、サーバーをサンプリングするのではなくすべてのサーバーをクエリすることによって、偽の空のレスポンスが排除されます。ロングポーリングを有効にするには、受信したメッセージについてゼロ以外の待機時間を指定します。詳細については、「SQS ロングポーリング」を参照してください。

以下の例では、次の方法を示しています。

- を使用して、Amazon SQS キューの属性を設定してロングポーリングを有効にしますSetQueueAttributes。
- を使用して、ロングポーリングで 1 つ以上のメッセージを取得しますReceiveMessage。
- を使用してロングポーリングキューを作成しますCreateQueue。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

キューの属性を設定してロングポーリングを有効にする

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

サンプルコード

```
$queueUrl = "QUEUE_URL";
```

```
$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->setQueueAttributes([
        'Attributes' => [
            'ReceiveMessageWaitTimeSeconds' => 20
        ],
        'QueueUrl' => $queueUrl, // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

ロングポーリングでメッセージを取得する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

サンプルコード

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->receiveMessage([
```

```
        'AttributeNames' => ['SentTimestamp'],
        'MaxNumberOfMessages' => 1,
        'MessageAttributeNames' => ['All'],
        'QueueUrl' => $queueUrl, // REQUIRED
        'WaitTimeSeconds' => 20,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

ロングポーリングでキューを作成する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

サンプルコード

```
$queueName = "QUEUE_NAME";


$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->createQueue([
        'QueueName' => $queueName,
        'Attributes' => [
            'ReceiveMessageWaitTimeSeconds' => 20
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 による Amazon SQS での可視性タイムアウトの管理

可視性タイムアウトは、Amazon SQS によって他の消費コンポーネントがそのメッセージを受信および処理できなくなる期間です。詳細については、「可視性タイムアウト」を参照してください。

以下の例では、次の方法を示しています。

- を使用して、キュー内の指定されたメッセージの可視性タイムアウトを新しい値に変更しますChangeMessageVisibilityBatch。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

複数メッセージの可視性タイムアウトの変更

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

サンプルコード

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
```

```
]);

try {
    $result = $client->receiveMessage(array(
        'AttributeNames' => ['SentTimestamp'],
        'MaxNumberOfMessages' => 10,
        'MessageAttributeNames' => ['All'],
        'QueueUrl' => $queueUrl, // REQUIRED
    ));
    $messages = $result->get('Messages');
    if ($messages != null) {
        $entries = array();
        for ($i = 0; $i < count($messages); $i++) {
            $entries[] = [
                'Id' => 'unique_is_msg' . $i, // REQUIRED
                'ReceiptHandle' => $messages[$i]['ReceiptHandle'], // REQUIRED
                'VisibilityTimeout' => 3600
            ];
        }
        $result = $client->changeMessageVisibilityBatch([
            'Entries' => $entries,
            'QueueUrl' => $queueUrl
        ]);

        var_dump($result);
    } else {
        echo "No messages in queue \n";
    }
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 を使用した Amazon SQS でのメッセージの送受信

Amazon SQS メッセージの詳細については、「Service Quotas ユーザーガイド」の「SQS キューへのメッセージの送信」および「SQS キューからのメッセージの受信および削除」を参照してください。

以下の例では、次の方法を示しています。

- を使用して、指定されたキューにメッセージを配信しますSendMessage。

- を使用して、指定されたキューから 1 つ以上のメッセージ (最大 10 件) を取得しますReceiveMessage。
- を使用してキューからメッセージを削除しますDeleteMessage。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

メッセージの送信

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

サンプルコード

```
$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

$params = [
    'DelaySeconds' => 10,
    'MessageAttributes' => [
        "Title" => [
            'DataType' => "String",
            'StringValue' => "The Hitchhiker's Guide to the Galaxy"
        ],
        "Author" => [
            'DataType' => "String",
            'StringValue' => "Douglas Adams."
        ],
        "WeeksOn" => [
```

```
            'DataType' => "Number",
            'StringValue' => "6"
        ]
    ],
    'MessageBody' => "Information about current NY Times fiction bestseller for week of
 12/11/2016.",
    'QueueUrl' => 'QUEUE_URL'
];

try {
    $result = $client->sendMessage($params);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

メッセージの受信および削除

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

サンプルコード

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->receiveMessage([
        'AttributeNames' => ['SentTimestamp'],
        'MaxNumberOfMessages' => 1,
        'MessageAttributeNames' => ['All'],
```

```
        'QueueUrl' => $queueUrl, // REQUIRED
        'WaitTimeSeconds' => 0,
    ]);
    if (!empty($result->get('Messages'))) {
        var_dump($result->get('Messages')[0]);
        $result = $client->deleteMessage([
            'QueueUrl' => $queueUrl, // REQUIRED
            'ReceiptHandle' => $result->get('Messages')[0]['ReceiptHandle'] // REQUIRED
        ]);
    } else {
        echo "No messages in queue. \n";
    }
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 による Amazon SQS でのデッドレターキューの使用

デッドレターキューは、正常に処理できないメッセージの送信先として他の (送信元) キューが使用できるキューです。これらのメッセージは、処理が成功しなかった理由を判断するためにデッドレターキューに分離できます。デッドレターキューにメッセージを送信する各ソースキューを、個別に設定する必要があります。1 つのデッドレターキューを複数のキューの送信先とすることができます。

詳細については、「SQS デッドレターキューの使用」を参照してください。

以下の例では、次の方法を示しています。

- を使用してデッドレターキューを有効にしますSetQueueAttributes。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

デッドレターキューを有効にする

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

サンプルコード

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->setQueueAttributes([
        'Attributes' => [
            'RedrivePolicy' => "{\"deadLetterTargetArn\":\"DEAD_LETTER_QUEUE_ARN\",
\"maxReceiveCount\":\"10\"}"
        ],
        'QueueUrl' => $queueUrl // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP バージョン 3 による Amazon SQS でのキューの使用

Amazon SQS キューについては、「SQS キューの操作」を参照してください。

以下の例では、次の方法を示しています。

- を使用してキューのリストを返しますListQueues。
- を使用して新しいキューを作成しますCreateQueue。
- を使用して既存のキューの URL を返しますGetQueueUrl。
- を使用して、指定されたキューを削除しますDeleteQueue。

のすべてのサンプルコードAWS SDK for PHPは、 にあります GitHub。

認証情報

サンプルコードを実行する前に、AWS の認証情報を設定します (認証情報 を参照)。AWS SDK for PHP からのインポート (基本的な使用法 を参照)。

キューのリストを取得する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

サンプルコード

```
$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->listQueues();
    foreach ($result->get('QueueUrls') as $queueUrl) {
        echo "$queueUrl\n";
    }
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

キューを作成する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
use Aws\Sqs\SqsClient;
```

サンプルコード

```
$queueName = "SQS_QUEUE_NAME";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->createQueue([
        'QueueName' => $queueName,
        'Attributes' => [
            'DelaySeconds' => 5,
            'MaximumMessageSize' => 4096, // 4 KB
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

キューの URL を取得する

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

サンプルコード

```
$queueName = "SQS_QUEUE_NAME";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->getQueueUrl([
        'QueueName' => $queueName // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

キューの削除

インポート

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

サンプルコード

```

$queueUrl = "SQS_QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->deleteQueue([
        'QueueUrl' => $queueUrl // REQUIRED
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Amazon EventBridge グローバルエンドポイントにイベントを送信する

Amazon EventBridge グローバルエンドポイントを使用して、イベント駆動型アプリケーションの可用性と信頼性を向上させることができます。

EventBridge グローバルエンドポイントを設定したら、 SDK for PHP を使用してグローバルエンドポイントにイベントを送信できます。

> **Important**
>
> SDK for PHP で EventBridge グローバルエンドポイントを使用するには、PHP 環境に AWS Common Runtime (AWS CRT) 拡張機能がインストールされている必要があります。

次の例では、 の PutEventsメソッドを使用してEventBridgeClient、1 つのイベントを EventBridge グローバルエンドポイントに送信します。

```
<?php
/* Send a single event to an existing Amazon EventBridge global endpoint. */
require '../vendor/autoload.php';

use Aws\EventBridge\EventBridgeClient;

$evClient = new EventBridgeClient([
    'region' => 'us-east-1'
]);

$endpointId = 'xxxx123456.xxx';  // Existing EventBridge global endpointId.
$eventBusName = 'default';       // Existing event bus in the us-east-1 Region.

$event =  [
    'Source' => 'my-php-app',
    'DetailType' => 'test',
```

```
    'Detail' => json_encode(['foo' => 'bar']),
    'Time' => new DateTime(),
    'Resources' => ['php-script'],
    'EventBusName' => $eventBusName,
    'TraceHeader' => 'test'
];

$result = $evClient->putEvents([
    'EndpointId' => $endpointId,
    'Entries' => [$event]
]);
```

このブログ記事には、 EventBridge グローバルエンドポイントに関する詳細情報が含まれています。

SDK PHPコード例の

このトピックのコード例は、 AWS SDK for PHP で を使用する方法を示しています AWS。

「基本」は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

サービス

- API SDKで を使用するゲートウェイの例 PHP
- SDK に を使用する Aurora の例 PHP
- SDK の を使用した Auto Scaling の例 PHP
- SDK に を使用する Amazon Bedrock の例 PHP
- SDK の を使用した Amazon Bedrock ランタイムの例 PHP
- SDK で を使用する Amazon DocumentDB の例 PHP
- の を使用する DynamoDB SDK の例 PHP
- AWS Glue SDKで を使用する の例 PHP
- IAM SDKで を使用する の例 PHP
- SDK で を使用する Kinesis の例 PHP
- SDK で を使用する Lambda の例 PHP
- SDK に を使用する Amazon RDSの例 PHP
- SDK の を使用した Amazon RDS Data Service の例 PHP
- SDK に を使用する Amazon Rekognition の例 PHP
- で を使用する Amazon S3 SDK の例 PHP
- SDK で を使用する Amazon SESの例 PHP
- SDK に を使用する Amazon SNSの例 PHP
- SDK に を使用する Amazon SQSの例 PHP

API SDKで を使用するゲートウェイの例 PHP

次のコード例は、APIGateway AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

アクション

`GetBasePathMapping`

次のコード例は、`GetBasePathMapping` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ////////////////////////////////////////////////////////////////////////////
```

```
 * Purpose: Gets the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 *
 * Returns: The base path mapping, if available; otherwise, the error message.
 * ////////////////////////////////////////////////////////////////////////// */

function getBasePathMapping($apiGatewayClient, $basePath, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
        ]);
        return 'The base path mapping\'s effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function getsTheBasePathMapping()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo getBasePathMapping($apiGatewayClient, '(none)', 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// getsTheBasePathMapping();
```

- API 詳細については、「 リファレンスGetBasePathMapping」の「」を参照してください。AWS SDK for PHP API

`ListBasePathMappings`

次の例は、ListBasePathMappings を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ////////////////////////////////////////////////////////////////////////
 * Purpose: Lists the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $domainName: The custom domain name for the base path mappings.
 *
 * Returns: Information about the base path mappings, if available;
 * otherwise, the error message.
 * //////////////////////////////////////////////////////////////////////// */

function listBasePathMappings($apiGatewayClient, $domainName)
{
```

```
    try {
        $result = $apiGatewayClient->getBasePathMappings([
            'domainName' => $domainName
        ]);
        return 'The base path mapping(s) effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function listTheBasePathMappings()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo listBasePathMappings($apiGatewayClient, 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// listTheBasePathMappings();
```

- API 詳細については、「 リファレンスListBasePathMappings」の「」を参照してください。 AWS SDK for PHP API

`UpdateBasePathMapping`

次の例は、UpdateBasePathMapping を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ////////////////////////////////////////////////////////////////////////
 *
 * Purpose: Updates the base path mapping for a custom domain name
 * in Amazon API Gateway.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 * - $patchOperations: The base path update operations to apply.
 *
 * Returns: Information about the updated base path mapping, if available;
 * otherwise, the error message.
 * //////////////////////////////////////////////////////////////////////// */

function updateBasePathMapping(
    $apiGatewayClient,
    $basePath,
    $domainName,
    $patchOperations
) {
    try {
        $result = $apiGatewayClient->updateBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
            'patchOperations' => $patchOperations
        ]);
        return 'The updated base path\'s URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function updateTheBasePathMapping()
```

```
{
    $patchOperations = array([
        'op' => 'replace',
        'path' => '/stage',
        'value' => 'stage2'
    ]);

    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo updateBasePathMapping(
        $apiGatewayClient,
        '(none)',
        'example.com',
        $patchOperations
    );
}

// Uncomment the following line to run this code in an AWS account.
// updateTheBasePathMapping();
```

- API 詳細については、「 リファレンスUpdateBasePathMapping」の「」を参照してください。 AWS SDK for PHP API

シナリオ

サーバーレスアプリケーションを作成して写真の管理

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

PHP に関する SDK

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

SDK に を使用する Aurora の例 PHP

次のコード例は、Aurora AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- シナリオ

シナリオ

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信するウェブアプリケーションを作成する方法を示しています。

PHP に関する SDK

を使用して、Amazon RDS データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon) を使用してレポートを E メールで送信するウェブアプリケーション AWS SDK for

PHP を作成する方法を示しますSES。この例では、React.js で構築されたフロントエンドを使用してRESTfulPHPバックエンドとやり取りします。

- React.js ウェブアプリケーションを AWS サービスと統合します。
- Amazon RDSテーブル内の項目を一覧表示、追加、更新、削除します。
- Amazon を使用して、フィルタリングされた作業項目の E メールレポートを送信しますSES。
- 付属の AWS CloudFormation スクリプトを使用してサンプルリソースをデプロイおよび管理します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

SDK の を使用した Auto Scaling の例 PHP

次のコード例は、Auto Scaling AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

「基本」は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

開始方法

こんにちは、Auto Scaling

次のコード例は、Auto Scaling の使用を開始する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
public function helloService()
{
    $autoScalingClient = new AutoScalingClient([
        'region' => 'us-west-2',
        'version' => 'latest',
        'profile' => 'default',
    ]);

    $groups = $autoScalingClient->describeAutoScalingGroups([]);
    var_dump($groups);
}
```

- API 詳細については、「 リファレンスDescribeAutoScalingGroups」の「」を参照してください。 AWS SDK for PHP API

トピック

- 基礎
- アクション

基礎

基本を学ぶ

次のコードサンプルは、以下の操作方法を示しています。

- 起動テンプレートとアベイラビリティーゾーンを使用して Amazon EC2 Auto Scaling グループを作成し、実行中のインスタンスに関する情報を取得します。
- Amazon CloudWatch メトリクス収集を有効にします。
- グループの希望するキャパシティを更新し、インスタンスが起動するのを待ちます。

- グループ内の最も古いインスタンスを削除します。
- ユーザーのリクエストやキャパシティの変更に応じて発生するスケーリングアクティビティを一覧表示します。
- CloudWatch メトリクスの統計情報を取得し、リソースをクリーンアップします。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
namespace AutoScaling;

use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;

class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
    protected array $role;

    public function runExample()
    {
        echo("\n");
        echo("-------------------------------------\n");
        print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
 PHP!\n");
        echo("-------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
```

```
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $this->autoScalingClient = new AutoScalingClient($clientArgs);
        $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
        $this->cloudWatchClient = new CloudWatchClient($clientArgs);

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        /**
         * Step 0: Create an EC2 launch template that you'll use to create an Auto
 Scaling group.
         */
        $this->ec2Client = new EC2Client($clientArgs);
        $this->templateName = "example_launch_template_$uniqid";
        $instanceType = "t1.micro";
        $amiId = "ami-0ca285d4c2cda3300";
        $launchTemplate = $this->ec2Client->createLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
            ]
        );

        /**
         * Step 1: CreateAutoScalingGroup: pass it the launch template you created
 in step 0.
         */
        $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];

        $this->autoScalingGroupName = "demoAutoScalingGroupName_$uniqid";
        $minSize = 1;
        $maxSize = 1;
        $launchTemplateId = $launchTemplate['LaunchTemplate']['LaunchTemplateId'];
        $this->autoScalingService->createAutoScalingGroup(
            $this->autoScalingGroupName,
```

```
            $availabilityZones,
            $minSize,
            $maxSize,
            $launchTemplateId
        );

        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);

        /**
         * Step 2: DescribeAutoScalingInstances: show that one instance has
 launched.
         */
        $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
        $instances = $this->autoScalingService-
>describeAutoScalingInstances($instanceIds);
        echo "The Auto Scaling group {$this->autoScalingGroupName} was created
 successfully.\n";
        echo count($instances['AutoScalingInstances']) . " instances were created
 for the group.\n";
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
 number of instances for the group.\n";

        /**
         * Step 3: EnableMetricsCollection: enable all metrics or a subset.
         */
        $this->autoScalingService->enableMetricsCollection($this-
>autoScalingGroupName, "1Minute");

        /**
         * Step 4: UpdateAutoScalingGroup: update max size to 3.
         */
        echo "Updating the max number of instances to 3.\n";
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

        /**
         * Step 5: DescribeAutoScalingGroups: show the current state of the group.
         */
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
```

```
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
        echo " is the updated max number of instances for the group.\n";

        $limits = $this->autoScalingService->describeAccountLimits();
        echo "Here are your account limits:\n";
        echo "MaxNumberOfAutoScalingGroups:
 {$limits['MaxNumberOfAutoScalingGroups']}\n";
        echo "MaxNumberOfLaunchConfigurations:
 {$limits['MaxNumberOfLaunchConfigurations']}\n";
        echo "NumberOfAutoScalingGroups: {$limits['NumberOfAutoScalingGroups']}\n";
        echo "NumberOfLaunchConfigurations:
 {$limits['NumberOfLaunchConfigurations']}\n";

        /**
         * Step 6: SetDesiredCapacity: set desired capacity to 2.
         */
        $this->autoScalingService->setDesiredCapacity($this->autoScalingGroupName,
 2);
        sleep(10); // Wait for the group to start processing the request.
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);

        /**
         * Step 7: DescribeAutoScalingInstances: show that two instances are
 launched.
         */
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
 instances in the group.
         */
```

```
        $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
        do {
            sleep(10);
            $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
        } while (count($instances['AutoScalingInstances']) > 0);
        do {
            sleep(10);
            $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
            $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
        } while (count($instances) < 2);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 9: DescribeScalingActivities: list the scaling activities that have
 occurred for the group so far.
         */
        $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
        echo "We found " . count($activities['Activities']) . " activities.\n";
        foreach ($activities['Activities'] as $activity) {
            echo "{$activity['ActivityId']} - {$activity['StartTime']} -
 {$activity['Description']}\n";
        }

        /**
         * Step 10: Use the Amazon CloudWatch API to get and show some metrics
 collected for the group.
         */
        $metricsNamespace = 'AWS/AutoScaling';
```

```
        $metricsDimensions = [
            [
                'Name' => 'AutoScalingGroupName',
                'Value' => $autoScalingGroup['AutoScalingGroupName'],
            ],
        ];
        $metrics = $this->cloudWatchClient->listMetrics(
            [
            'Dimensions' => $metricsDimensions,
            'Namespace' => $metricsNamespace,
            ]
        );
        foreach ($metrics['Metrics'] as $metric) {
            $timespan = 5;
            if ($metric['MetricName'] != 'GroupTotalCapacity' &&
 $metric['MetricName'] != 'GroupMaxSize') {
                continue;
            }
            echo "Over the last $timespan minutes, {$metric['MetricName']} recorded:
\n";
            $stats = $this->cloudWatchClient->getMetricStatistics(
                [
                'Dimensions' => $metricsDimensions,
                'EndTime' => time(),
                'StartTime' => time() - (5 * 60),
                'MetricName' => $metric['MetricName'],
                'Namespace' => $metricsNamespace,
                'Period' => 60,
                'Statistics' => ['Sum'],
                ]
            );
            foreach ($stats['Datapoints'] as $stat) {
                echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
            }
        }

        return $instances;
    }

    public function cleanUp()
    {
        /**
         * Step 11: DisableMetricsCollection: disable all metrics.
         */
```

```
        $this->autoScalingService->disableMetricsCollection($this-
>autoScalingGroupName);

        /**
         * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
 instances.
         * - UpdateAutoScalingGroup with MinSize=0
         * - TerminateInstanceInAutoScalingGroup for each instance,
         *     specify ShouldDecrementDesiredCapacity=True. Wait for instances to
 stop.
         * - Now you can delete the group.
         */
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MinSize' => 0]);
        $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this-
>autoScalingGroupName);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $this->autoScalingService->deleteAutoScalingGroup($this-
>autoScalingGroupName);

        /**
         * Step 13: Delete launch template.
         */
        $this->ec2Client->deleteLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            ]
        );
    }

    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
}
```

- API 詳細については、「 AWS SDK for PHP APIリファレンス」の以下のトピックを参照してください。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

アクション

`CreateAutoScalingGroup`

次の例は、`CreateAutoScalingGroup` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    public function createAutoScalingGroup(
        $autoScalingGroupName,
        $availabilityZones,
        $minSize,
        $maxSize,
        $launchTemplateId
    ) {
```

```
        return $this->autoScalingClient->createAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'AvailabilityZones' => $availabilityZones,
            'MinSize' => $minSize,
            'MaxSize' => $maxSize,
            'LaunchTemplate' => [
                'LaunchTemplateId' => $launchTemplateId,
            ],
        ]);
    }
```

- API 詳細については、「 リファレンスCreateAutoScalingGroup」の「」を参照してください。AWS SDK for PHP API

`DeleteAutoScalingGroup`

次のコード例は、DeleteAutoScalingGroup を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    public function deleteAutoScalingGroup($autoScalingGroupName)
    {
        return $this->autoScalingClient->deleteAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'ForceDelete' => true,
        ]);
    }
```

- API 詳細については、「 リファレンスDeleteAutoScalingGroup」の「」を参照してください。AWS SDK for PHP API

DescribeAutoScalingGroups

次のコード例は、DescribeAutoScalingGroups を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
public function describeAutoScalingGroups($autoScalingGroupNames)
{
    return $this->autoScalingClient->describeAutoScalingGroups([
        'AutoScalingGroupNames' => $autoScalingGroupNames
    ]);
}
```

- API 詳細については、「 リファレンスDescribeAutoScalingGroups」の「」を参照してください。 AWS SDK for PHP API

DescribeAutoScalingInstances

次のコード例は、DescribeAutoScalingInstances を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
public function describeAutoScalingInstances($instanceIds)
{
    return $this->autoScalingClient->describeAutoScalingInstances([
        'InstanceIds' => $instanceIds
    ]);
```

```
    }
```

- API 詳細については、「 リファレンスDescribeAutoScalingInstances」の「」を参照してください。 AWS SDK for PHP API

`DescribeScalingActivities`

次の例は、`DescribeScalingActivities` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
public function describeScalingActivities($autoScalingGroupName)
{
    return $this->autoScalingClient->describeScalingActivities([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- API 詳細については、「 リファレンスDescribeScalingActivities」の「」を参照してください。 AWS SDK for PHP API

`DisableMetricsCollection`

次の例は、`DisableMetricsCollection` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    public function disableMetricsCollection($autoScalingGroupName)
    {
        return $this->autoScalingClient->disableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- API 詳細については、「 リファレンスDisableMetricsCollection」の「」を参照してください。AWS SDK for PHP API

`EnableMetricsCollection`

次の例は、EnableMetricsCollection を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    public function enableMetricsCollection($autoScalingGroupName, $granularity)
    {
        return $this->autoScalingClient->enableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'Granularity' => $granularity,
        ]);
    }
```

- API 詳細については、「 リファレンスEnableMetricsCollection」の「」を参照してください。AWS SDK for PHP API

`SetDesiredCapacity`

次のコード例は、SetDesiredCapacity を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
{
    return $this->autoScalingClient->setDesiredCapacity([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'DesiredCapacity' => $desiredCapacity,
    ]);
}
```

- API 詳細については、「 リファレンスSetDesiredCapacity」の「」を参照してください。AWS SDK for PHP API

`TerminateInstanceInAutoScalingGroup`

次のコード例は、TerminateInstanceInAutoScalingGroup を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
public function terminateInstanceInAutoScalingGroup(
    $instanceId,
    $shouldDecrementDesiredCapacity = true,
    $attempts = 0
) {
    try {
        return $this->autoScalingClient->terminateInstanceInAutoScalingGroup([
            'InstanceId' => $instanceId,
```

```
                'ShouldDecrementDesiredCapacity' => $shouldDecrementDesiredCapacity,
            ]);
        } catch (AutoScalingException $exception) {
            if ($exception->getAwsErrorCode() == "ScalingActivityInProgress" &&
$attempts < 5) {
                error_log("Cannot terminate an instance while it is still pending.
Waiting then trying again.");
                sleep(5 * (1 + $attempts));
                return $this->terminateInstanceInAutoScalingGroup(
                    $instanceId,
                    $shouldDecrementDesiredCapacity,
                    ++$attempts
                );
            } else {
                throw $exception;
            }
        }
    }
```

- API 詳細については、「 リファレンスTerminateInstanceInAutoScalingGroup」の「」を参照してください。 AWS SDK for PHP API

UpdateAutoScalingGroup

次の例は、UpdateAutoScalingGroup を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    public function updateAutoScalingGroup($autoScalingGroupName, $args)
    {
        if (array_key_exists('MaxSize', $args)) {
            $maxSize = ['MaxSize' => $args['MaxSize']];
        } else {
            $maxSize = [];
```

```
        }
        if (array_key_exists('MinSize', $args)) {
            $minSize = ['MinSize' => $args['MinSize']];
        } else {
            $minSize = [];
        }
        $parameters = ['AutoScalingGroupName' => $autoScalingGroupName];
        $parameters = array_merge($parameters, $minSize, $maxSize);
        return $this->autoScalingClient->updateAutoScalingGroup($parameters);
    }
```

- API 詳細については、「 リファレンスUpdateAutoScalingGroup」の「」を参照してください。AWS SDK for PHP API

SDK に を使用する Amazon Bedrock の例 PHP

次のコード例は、Amazon Bedrock AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- アクション

アクション

`ListFoundationModels`

次のコード例は、`ListFoundationModels` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

利用可能な Amazon Bedrock 基盤モデルを一覧表示します。

```
public function listFoundationModels()
{
    $result = $this->bedrockClient->listFoundationModels();
    return $result;
}
```

- API 詳細については、「 リファレンスListFoundationModels」の「」を参照してください。AWS SDK for PHP API

SDK の を使用した Amazon Bedrock ランタイムの例 PHP

次のコード例は、Amazon Bedrock ランタイム AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- シナリオ
- AI21 ラボ Jurassic-2
- Amazon Titan Image Generator
- Anthropic Claude
- メタラマ
- Stable Diffusion

シナリオ

Amazon Bedrock で複数の基盤モデルを呼び出す

次のコード例は、Amazon Bedrock のさまざまな大規模言語モデル (LLMs) でプロンプトを準備して送信する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

Amazon Bedrock LLMsで複数の を呼び出します。

```
namespace BedrockRuntime;

class GettingStartedWithBedrockRuntime
{
    protected BedrockRuntimeService $bedrockRuntimeService;

    public function runExample()
    {
        echo "\n";
        echo "---------------------------------------------------------------------
\n";
        echo "Welcome to the Amazon Bedrock Runtime getting started demo using PHP!
\n";
        echo "---------------------------------------------------------------------
\n";

        $clientArgs = [
            'region' => 'us-east-1',
            'version' => 'latest',
            'profile' => 'default',
        ];

        $bedrockRuntimeService = new BedrockRuntimeService($clientArgs);

        $prompt = 'In one paragraph, who are you?';
```

```
        echo "\nPrompt: " . $prompt;

        echo "\n\nAnthropic Claude:";
        echo $bedrockRuntimeService->invokeClaude($prompt);

        echo "\n\nAI21 Labs Jurassic-2: ";
        echo $bedrockRuntimeService->invokeJurassic2($prompt);

        echo "\n\nMeta Llama 2 Chat: ";
        echo $bedrockRuntimeService->invokeLlama2($prompt);

        echo
"\n----------------------------------------------------------------------\n";

        $image_prompt = 'stylized picture of a cute old steampunk robot';

        echo "\nImage prompt: " . $image_prompt;

        echo "\n\nStability.ai Stable Diffusion XL:\n";
        $diffusionSeed = rand(0, 4294967295);
        $style_preset = 'photographic';
        $base64 = $bedrockRuntimeService->invokeStableDiffusion($image_prompt,
$diffusionSeed, $style_preset);
        $image_path = $this->saveImage($base64, 'stability.stable-diffusion-xl');
        echo "The generated images have been saved to $image_path";

        echo "\n\nAmazon Titan Image Generation:\n";
        $titanSeed = rand(0, 2147483647);
        $base64 = $bedrockRuntimeService->invokeTitanImage($image_prompt,
$titanSeed);
        $image_path = $this->saveImage($base64, 'amazon.titan-image-generator-v1');
        echo "The generated images have been saved to $image_path";
    }

    private function saveImage($base64_image_data, $model_id): string
    {
        $output_dir = "output";

        if (!file_exists($output_dir)) {
            mkdir($output_dir);
        }

        $i = 1;
```

```
        while (file_exists("$output_dir/$model_id" . '_' . "$i.png")) {
            $i++;
        }

        $image_data = base64_decode($base64_image_data);

        $file_path = "$output_dir/$model_id" . '_' . "$i.png";

        $file = fopen($file_path, 'wb');
        fwrite($file, $image_data);
        fclose($file);

        return $file_path;
    }
}
```

- API 詳細については、「 AWS SDK for PHP APIリファレンス」の以下のトピックを参照してください。
  - InvokeModel
  - InvokeModelWithResponseStream

AI21 ラボ Jurassic-2

InvokeModel

次のコード例は、Invoke Model を使用して AI21 Labs Jurassic-2 にテキストメッセージを送信する方法を示していますAPI。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

モデル呼び出しを使用してテキストメッセージAPIを送信します。

```
    public function invokeJurassic2($prompt)
    {
        # The different model providers have individual request and response
 formats.
        # For the format, ranges, and default values for AI21 Labs Jurassic-2, refer
 to:
        # https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
jurassic2.html

        $completion = "";

        try {
            $modelId = 'ai21.j2-mid-v1';

            $body = [
                'prompt' => $prompt,
                'temperature' => 0.5,
                'maxTokens' => 200,
            ];

            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);

            $response_body = json_decode($result['body']);

            $completion = $response_body->completions[0]->data->text;
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $completion;
    }
```

- API 詳細については、「 リファレンスInvokeModel」の「」を参照してください。 AWS SDK for PHP API

Amazon Titan Image Generator

InvokeModel

次のコード例は、Amazon Bedrock で Amazon Titan Image を呼び出してイメージを生成する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

Amazon Titan Image Generator を使用してイメージを作成します。

```
    public function invokeTitanImage(string $prompt, int $seed)
    {
        # The different model providers have individual request and response
 formats.
        # For the format, ranges, and default values for Titan Image models refer
 to:
        # https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html

        $base64_image_data = "";

        try {
            $modelId = 'amazon.titan-image-generator-v1';

            $request = json_encode([
                'taskType' => 'TEXT_IMAGE',
                'textToImageParams' => [
                    'text' => $prompt
                ],
                'imageGenerationConfig' => [
                    'numberOfImages' => 1,
                    'quality' => 'standard',
                    'cfgScale' => 8.0,
                    'height' => 512,
                    'width' => 512,
```

```
                'seed' => $seed
            ]
        ]);

        $result = $this->bedrockRuntimeClient->invokeModel([
            'contentType' => 'application/json',
            'body' => $request,
            'modelId' => $modelId,
        ]);

        $response_body = json_decode($result['body']);

        $base64_image_data = $response_body->images[0];
    } catch (Exception $e) {
        echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
    }

    return $base64_image_data;
}
```

- API 詳細については、「 リファレンスInvokeModel」の「」を参照してください。 AWS SDK for PHP API

Anthropic Claude

InvokeModel

次のコード例は、Invoke Model を使用して Anthropic Claude にテキストメッセージを送信する方法を示していますAPI。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

Anthropic Claude 2 基盤モデルを呼び出して、テキストを生成します。

```
    public function invokeClaude($prompt)
    {
        # The different model providers have individual request and response
 formats.
        # For the format, ranges, and default values for Anthropic Claude, refer to:
        # https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
claude.html

        $completion = "";

        try {
            $modelId = 'anthropic.claude-v2';

            # Claude requires you to enclose the prompt as follows:
            $prompt = "\n\nHuman: {$prompt}\n\nAssistant:";

            $body = [
                'prompt' => $prompt,
                'max_tokens_to_sample' => 200,
                'temperature' => 0.5,
                'stop_sequences' => ["\n\nHuman:"],
            ];

            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);

            $response_body = json_decode($result['body']);

            $completion = $response_body->completion;
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $completion;
    }
```

- API 詳細については、「 リファレンスInvokeModel」の「」を参照してください。 AWS SDK for PHP API

メタラマ

InvokeModel: ラマ 2

次のコード例は、モデル を呼び出す を使用して Meta Llama 2 にテキストメッセージを送信する方法を示していますAPI。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

モデル呼び出しを使用してテキストメッセージAPIを送信します。

```
    public function invokeLlama2($prompt)
    {
        # The different model providers have individual request and response
 formats.
        # For the format, ranges, and default values for Meta Llama 2 Chat, refer
 to:
        # https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html

        $completion = "";

        try {
            $modelId = 'meta.llama2-13b-chat-v1';

            $body = [
                'prompt' => $prompt,
                'temperature' => 0.5,
                'max_gen_len' => 512,
            ];

            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);
```

```
            $response_body = json_decode($result['body']);

            $completion = $response_body->generation;
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $completion;
    }
```

- API 詳細については、「 リファレンスInvokeModel」の「」を参照してください。 AWS SDK for PHP API

Stable Diffusion

InvokeModel

次のコード例は、Amazon Bedrock で Stability.ai Stable Diffusion XL を呼び出してイメージを生成する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

Stable Diffusion を使用してイメージを作成します。

```
    public function invokeStableDiffusion(string $prompt, int $seed, string
 $style_preset)
    {
        # The different model providers have individual request and response
 formats.
        # For the format, ranges, and available style_presets of Stable Diffusion
 models refer to:
        # https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
stability-diffusion.html
```

```
        $base64_image_data = "";

        try {
            $modelId = 'stability.stable-diffusion-xl';

            $body = [
                'text_prompts' => [
                    ['text' => $prompt]
                ],
                'seed' => $seed,
                'cfg_scale' => 10,
                'steps' => 30
            ];

            if ($style_preset) {
                $body['style_preset'] = $style_preset;
            }

            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);

            $response_body = json_decode($result['body']);

            $base64_image_data = $response_body->artifacts[0]->base64;
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $base64_image_data;
    }
```

- API 詳細については、「 リファレンスInvokeModel」の「」を参照してください。 AWS SDK for PHP API

SDK で を使用する Amazon DocumentDB の例 PHP

次のコード例は、Amazon DocumentDB AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- サーバーレスサンプル

サーバーレスサンプル

Amazon DocumentDB トリガーから Lambda 関数を呼び出す

次のコード例は、DocumentDB 変更ストリームからレコードを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示しています。関数は DocumentDB ペイロードを取得し、レコードの内容を口グ記録します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda で Amazon DocumentDB イベントを消費するPHP。

```
<?php

require __DIR__.'/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
```

```
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

の を使用する DynamoDB SDK の例 PHP

次のコード例は、DynamoDB AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

「基本」は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- 基礎
- アクション
- シナリオ
- サーバーレスサンプル

基礎

基本を学ぶ

次のコードサンプルは、以下の操作方法を示しています。

- 映画データを保持できるテーブルを作成する。
- テーブルに 1 つの映画を入れ、取得して更新する。
- サンプルJSONファイルからテーブルに映画データを書き込みます。
- 特定の年にリリースされた映画を照会する。
- 何年もの間にリリースされた映画をスキャンする。
- テーブルからムービーを削除し、テーブルを削除します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
namespace DynamoDb\Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;
use DynamoDb\DynamoDBService;
```

```
use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithDynamoDB
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDBService();

        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }

        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
```

```
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];
        $attributes = ["rating" =>
            [
                'AttributeName' => 'rating',
                'AttributeType' => 'N',
                'Value' => $rating,
            ],
            'plot' => [
                'AttributeName' => 'plot',
                'AttributeType' => 'S',
                'Value' => $plot,
            ]
        ];
        $service->updateItemAttributesByKey($tableName, $key, $attributes);
        echo "Movie added and updated.";

        $batch = json_decode(loadMovieData());
```

```
        $service->writeBatch($tableName, $batch);


        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
 {$movie['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
 $rating);

        $movie = $service->getItemByKey($tableName, $key);
        echo "Ok, you have rated {$movie['Item']['title']['S']} as a {$movie['Item']
['rating']['N']}\n";

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
```

```
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}
```

- API 詳細については、「 AWS SDK for PHP APIリファレンス」の以下のトピックを参照してください。
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query

- Scan
- UpdateItem

アクション

`BatchExecuteStatement`

次のコード例は、BatchExecuteStatement を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
```

```
                ],
            ],
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- API 詳細については、「 リファレンスBatchExecuteStatement」の「」を参照してください。AWS SDK for PHP API

`BatchWriteItem`

次のコード例は、BatchWriteItem を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    public function writeBatch(string $TableName, array $Batch, int $depth = 2)
    {
        if (--$depth <= 0) {
            throw new Exception("Max depth exceeded. Please try with fewer batch
items or increase depth.");
        }

        $marshal = new Marshaler();
        $total = 0;
        foreach (array_chunk($Batch, 25) as $Items) {
            foreach ($Items as $Item) {
                $BatchWrite['RequestItems'][$TableName][] = ['PutRequest' => ['Item'
=> $marshal->marshalItem($Item)]];
            }
            try {
                echo "Batching another " . count($Items) . " for a total of " .
($total += count($Items)) . " items!\n";
                $response = $this->dynamoDbClient->batchWriteItem($BatchWrite);
                $BatchWrite = [];
            } catch (Exception $e) {
                echo "uh oh...";
                echo $e->getMessage();
                die();
            }
            if ($total >= 250) {
                echo "250 movies is probably enough. Right? We can stop there.\n";
                break;
            }
        }
    }
```

- API 詳細については、「 リファレンスBatchWriteItem」の「」を参照してください。 AWS SDK for PHP API

CreateTable

次のコード例は、CreateTable を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

テーブルを作成します。

```
        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

    public function createTable(string $tableName, array $attributes)
    {
        $keySchema = [];
        $attributeDefinitions = [];
        foreach ($attributes as $attribute) {
            if (is_a($attribute, DynamoDBAttribute::class)) {
                $keySchema[] = ['AttributeName' => $attribute->AttributeName,
'KeyType' => $attribute->KeyType];
                $attributeDefinitions[] =
                    ['AttributeName' => $attribute->AttributeName, 'AttributeType'
=> $attribute->AttributeType];
            }
        }

        $this->dynamoDbClient->createTable([
            'TableName' => $tableName,
            'KeySchema' => $keySchema,
            'AttributeDefinitions' => $attributeDefinitions,
            'ProvisionedThroughput' => ['ReadCapacityUnits' => 10,
'WriteCapacityUnits' => 10],
```

```
        ]);
    }
```

- API 詳細については、「 リファレンスCreateTable」の「」を参照してください。 AWS SDK for PHP API

`DeleteItem`

次のコード例は、DeleteItem を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

    public function deleteItemByKey(string $tableName, array $key)
    {
        $this->dynamoDbClient->deleteItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- API 詳細については、「 リファレンスDeleteItem」の「」を参照してください。 AWS SDK for PHP API

`DeleteTable`

次の例は、`DeleteTable` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
public function deleteTable(string $TableName)
{
    $this->customWaiter(function () use ($TableName) {
        return $this->dynamoDbClient->deleteTable([
            'TableName' => $TableName,
        ]);
    });
}
```

- API 詳細については、「 リファレンスDeleteTable」の「」を参照してください。 AWS SDK for PHP API

`ExecuteStatement`

次のコード例は、`ExecuteStatement` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
$tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }
```

- API 詳細については、「 リファレンスExecuteStatement」の「」を参照してください。 AWS SDK for PHP API

`GetItem`

次の例は、`GetItem` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
{$movie['Item']['year']['N']}.\n";

    public function getItemByKey(string $tableName, array $key)
    {
        return $this->dynamoDbClient->getItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- API 詳細については、「 リファレンスGetItem」の「」を参照してください。 AWS SDK for PHP API

`ListTables`

次のコード例は、`ListTables` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
public function listTables($exclusiveStartTableName = "", $limit = 100)
{
    $this->dynamoDbClient->listTables([
        'ExclusiveStartTableName' => $exclusiveStartTableName,
        'Limit' => $limit,
    ]);
}
```

- API 詳細については、「 リファレンスListTables」の「」を参照してください。 AWS SDK for PHP API

`PutItem`

次のコード例は、PutItem を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    echo "What's the name of the last movie you watched?\n";
    while (empty($movieName)) {
        $movieName = testable_readline("Movie name: ");
    }
    echo "And what year was it released?\n";
    $movieYear = "year";
    while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
        $movieYear = testable_readline("Year released: ");
```

```
        }

        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

    public function putItem(array $array)
    {
        $this->dynamoDbClient->putItem($array);
    }
```

- API 詳細については、「 リファレンスPutItem」の「」を参照してください。 AWS SDK for PHP API

Query

次のコード例は、Query を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
```

```
        ];
        $result = $service->query($tableName, $birthKey);

    public function query(string $tableName, $key)
    {
        $expressionAttributeValues = [];
        $expressionAttributeNames = [];
        $keyConditionExpression = "";
        $index = 1;
        foreach ($key as $name => $value) {
            $keyConditionExpression .= "#" . array_key_first($value) . " = :v
$index,";
            $expressionAttributeNames["#" . array_key_first($value)] =
 array_key_first($value);
            $hold = array_pop($value);
            $expressionAttributeValues[":v$index"] = [
                array_key_first($hold) => array_pop($hold),
            ];
        }
        $keyConditionExpression = substr($keyConditionExpression, 0, -1);
        $query = [
            'ExpressionAttributeValues' => $expressionAttributeValues,
            'ExpressionAttributeNames' => $expressionAttributeNames,
            'KeyConditionExpression' => $keyConditionExpression,
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->query($query);
    }
```

- API 詳細については、「 リファレンス」の「クエリ」を参照してください。 AWS SDK for PHP API

Scan

次のコード例は、Scan を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    $yearsKey = [
        'Key' => [
            'year' => [
                'N' => [
                    'minRange' => 1990,
                    'maxRange' => 1999,
                ],
            ],
        ],
    ];
    $filter = "year between 1990 and 1999";
    echo "\nHere's a list of all the movies released in the 90s:\n";
    $result = $service->scan($tableName, $yearsKey, $filter);
    foreach ($result['Items'] as $movie) {
        $movie = $marshal->unmarshalItem($movie);
        echo $movie['title'] . "\n";
    }

public function scan(string $tableName, array $key, string $filters)
{
    $query = [
        'ExpressionAttributeNames' => ['#year' => 'year'],
        'ExpressionAttributeValues' => [
            ":min" => ['N' => '1990'],
            ":max" => ['N' => '1999'],
        ],
        'FilterExpression' => "#year between :min and :max",
        'TableName' => $tableName,
    ];
    return $this->dynamoDbClient->scan($query);
}
```

- API 詳細については、「 リファレンス」の「スキャン」を参照してください。 AWS SDK for PHP API

`UpdateItem`

次の例は、UpdateItem を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
$rating);

    public function updateItemAttributeByKey(
        string $tableName,
        array $key,
        string $attributeName,
        string $attributeType,
        string $newValue
    ) {
        $this->dynamoDbClient->updateItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
            'UpdateExpression' => "set #NV=:NV",
            'ExpressionAttributeNames' => [
                '#NV' => $attributeName,
            ],
            'ExpressionAttributeValues' => [
                ':NV' => [
                    $attributeType => $newValue
                ]
```

```
            ],
        ]);
    }
```

- API 詳細については、「 リファレンスUpdateItem」の「」を参照してください。 AWS SDK for PHP API

シナリオ

サーバーレスアプリケーションを作成して写真の管理

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

PHP に関する SDK

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PartiQL ステートメントのバッチを使用してテーブルにクエリを実行する

次のコードサンプルは、以下の操作方法を示しています。

- 複数のSELECTステートメントを実行して、項目のバッチを取得します。
- 複数のINSERTステートメントを実行して、項目のバッチを追加します。

- 複数のUPDATEステートメントを実行して、項目のバッチを更新します。
- 複数のDELETEステートメントを実行して、項目のバッチを削除します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithPartiQLBatch
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );
```

```
        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQLBatch($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];
```

```
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);
        echo "Movie added and updated.\n";

        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "\nThe movie {$movie['Responses'][0]['Item']['title']['S']}
        was released in {$movie['Responses'][0]['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Responses'][0]['Item']
['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "Okay, you have rated {$movie['Responses'][0]['Item']['title']['S']}
        as a {$movie['Responses'][0]['Item']['rating']['N']}\n";

        $service->deleteItemByPartiQLBatch($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
```

```
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
```

```
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
```

```
                ],
            ],
        ]);
    }
```

- API 詳細については、「 リファレンスBatchExecuteStatement」の「」を参照してください。AWS SDK for PHP API

PartiQL を使用してテーブルに対してクエリを実行する

次のコードサンプルは、以下の操作方法を示しています。

- SELECT ステートメントを実行して項目を取得します。
- INSERT ステートメントを実行して項目を追加します。
- UPDATE ステートメントを実行して項目を更新します。
- DELETE ステートメントを実行して項目を削除します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\testable_readline;
use function AwsUtilities\loadMovieData;

class GettingStartedWithPartiQL
{
    public function run()
    {
```

```
        echo("\n");
        echo("---------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
 PHP!\n");
        echo("---------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
```

```
        $service->insertItemByPartiQL($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);
        echo "Movie added and updated.\n";



        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "\nThe movie {$movie['Items'][0]['title']['S']} was released in
 {$movie['Items'][0]['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Items'][0]['title']['S']}?
\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
```

```
        $service->updateItemByPartiQL($statement, $parameters);

        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "Okay, you have rated {$movie['Items'][0]['title']['S']} as a
 {$movie['Items'][0]['rating']['N']}\n";

        $service->deleteItemByPartiQL($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
```

```
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
 $tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
```

```
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }
```

- API 詳細については、「 リファレンスExecuteStatement」の「」を参照してください。 AWS SDK for PHP API

サーバーレスサンプル

DynamoDB トリガーから Lambda 関数を呼び出す

次のコード例は、DynamoDB ストリームからレコードを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示しています。関数は DynamoDB ペイロードを取得し、レコードの内容をログ記録します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda で DynamoDB イベントを消費するPHP。

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends DynamoDbHandler
{
```

```
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

DynamoDB トリガーで Lambda 関数のバッチアイテムの失敗をレポートする

次のコード例は、DynamoDB ストリームからイベントを受信する Lambda 関数に部分的なバッチレスポンスを実装する方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用した Lambda での DynamoDB バッチアイテムの失敗のレポートPHP。

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
```

```
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

AWS Glue SDKで を使用する の例 PHP

次のコード例は、 AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS Glue。

「基本」は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

基礎

基本を学ぶ

次のコードサンプルは、以下の操作方法を示しています。

- パブリック Amazon S3 バケットをクロールし、 CSV形式のメタデータのデータベースを生成するクローラーを作成します。
- のデータベースとテーブルに関する情報を一覧表示します AWS Glue Data Catalog。
- S3 バケットからCSVデータを抽出し、データを変換して、JSONフォーマットされた出力を別の S3 バケットにロードするジョブを作成します。
- ジョブ実行に関する情報を一覧表示し、変換されたデータを表示してリソースをクリーンアップする。

詳細については、「チュートリアル: AWS Glue Studio の開始方法」を参照してください。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
namespace Glue;

use Aws\Glue\GlueClient;
```

```
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithGlue
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Glue getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
        $glueService = new GlueService($glueClient);
        $iamService = new IAMService();
        $crawlerName = "example-crawler-test-" . $uniqid;

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $databaseName = "doc-example-database-$uniqid";
        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
 $databaseName, $path);
        $glueService->startCrawler($crawlerName);

        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";
```

```
        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

        //Upload job script
        $s3client = new S3Client($clientArgs);
        $bucketName = "test-glue-bucket-" . $uniqid;
        $s3client->createBucket([
            'Bucket' => $bucketName,
            'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
        ]);

        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'run_job.py',
            'SourceFile' => __DIR__ . '/flight_etl_job_script.py'
        ]);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'setup_scenario_getting_started.yaml',
            'SourceFile' => __DIR__ . '/setup_scenario_getting_started.yaml'
        ]);

        $tables = $glueService->getTables($databaseName);

        $jobName = 'test-job-' . $uniqid;
        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

        $jobRuns = $glueService->getJobRuns($jobName);
```

```
$objects = $s3client->listObjects([
    'Bucket' => $bucketName,
])['Contents'];

foreach ($objects as $object) {
    echo $object['Key'] . "\n";
}

echo "Downloading " . $objects[1]['Key'] . "\n";
/** @var Stream $downloadObject */
$downloadObject = $s3client->getObject([
    'Bucket' => $bucketName,
    'Key' => $objects[1]['Key'],
])['Body']->getContents();
echo "Here is the first 1000 characters in the object.";
echo substr($downloadObject, 0, 1000);

$jobs = $glueService->listJobs();
echo "Current jobs:\n";
foreach ($jobs['JobNames'] as $jobsName) {
    echo "{$jobsName}\n";
}

echo "Delete the job.\n";
$glueClient->deleteJob([
    'JobName' => $job['Name'],
]);

echo "Delete the tables.\n";
foreach ($tables['TableList'] as $table) {
    $glueService->deleteTable($table['Name'], $databaseName);
}

echo "Delete the databases.\n";
$glueClient->deleteDatabase([
    'Name' => $databaseName,
]);

echo "Delete the crawler.\n";
$glueClient->deleteCrawler([
    'Name' => $crawlerName,
]);
```

```
        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        echo "Delete all objects in the bucket.\n";
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Delete the bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);

        echo "This job was brought to you by the number $uniqid\n";
    }
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;

use function PHPUnit\Framework\isEmpty;

class GlueService extends \AwsUtilities\AWSServiceClass
{
    protected GlueClient $glueClient;

    public function __construct($glueClient)
    {
        $this->glueClient = $glueClient;
    }

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
```

```
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
```

```
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }

    public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):
Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
    {
```

```
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }

    public function deleteCrawler($crawlerName)
    {
```

```
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
}
```

- API 詳細については、「 AWS SDK for PHP APIリファレンス」の以下のトピックを参照してください。
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

アクション

`CreateCrawler`

次のコード例は、`CreateCrawler` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }
```

- API 詳細については、「 リファレンスCreateCrawler」の「」を参照してください。 AWS SDK for PHP API

`CreateJob`

次のコード例は、CreateJob を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $jobName = 'test-job-' . $uniqid;

        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }
```

- API 詳細については、「 リファレンスCreateJob」の「」を参照してください。 AWS SDK for PHP API

`DeleteCrawler`

次の例は、`DeleteCrawler` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- API 詳細については、「 リファレンスDeleteCrawler」の「」を参照してください。 AWS SDK for PHP API

`DeleteDatabase`

次のコード例は、DeleteDatabase を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
```

```
        ]);

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- API 詳細については、「 リファレンスDeleteDatabase」の「」を参照してください。 AWS SDK for PHP API

`DeleteJob`

次の例は、DeleteJob を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- API 詳細については、「 リファレンスDeleteJob」の「」を参照してください。 AWS SDK for PHP API

DeleteTable

次の例は、DeleteTable を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }
```

- API 詳細については、「 リファレンスDeleteTable」の「」を参照してください。 AWS SDK for PHP API

GetCrawler

次のコード例は、GetCrawler を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }
```

- API 詳細については、「 リファレンスGetCrawler」の「」を参照してください。 AWS SDK for PHP API

`GetDatabase`

次の例は、GetDatabase を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        $databaseName = "doc-example-database-$uniqid";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
```

```
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }
```

- API 詳細については、「 リファレンスGetDatabase」の「」を参照してください。 AWS SDK for PHP API

GetJobRun

次の例は、GetJobRun を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        $jobName = 'test-job-' . $uniqid;

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
```

```
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }
```

- API 詳細については、「 リファレンスGetJobRun」の「」を参照してください。 AWS SDK for PHP API

`GetJobRuns`

次のコード例は、GetJobRuns を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        $jobName = 'test-job-' . $uniqid;

        $jobRuns = $glueService->getJobRuns($jobName);

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }
```

- API 詳細については、「 リファレンスGetJobRuns」の「」を参照してください。 AWS SDK for PHP API

GetTables

次のコード例は、GetTables を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }
```

- API 詳細については、「 リファレンスGetTables」の「」を参照してください。 AWS SDK for PHP API

ListJobs

次の例は、ListJobs を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        $jobs = $glueService->listJobs();
```

```
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }
```

- API 詳細については、「 リファレンスListJobs」の「」を参照してください。 AWS SDK for PHP API

`StartCrawler`

次のコード例は、StartCrawler を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $databaseName = "doc-example-database-$uniqid";
```

```
        $glueService->startCrawler($crawlerName);

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- API 詳細については、「 リファレンスStartCrawler」の「」を参照してください。 AWS SDK for PHP API

`StartJobRun`

次の例は、StartJobRun を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        $jobName = 'test-job-' . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

    public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):
Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
```

```
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }
```

- API 詳細については、「 リファレンスStartJobRun」の「」を参照してください。 AWS SDK for PHP API

IAM SDKで を使用する の例 PHP

次のコード例は、 AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示していますIAM。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- アクション
- シナリオ

アクション

`AttachRolePolicy`

次の例は、`AttachRolePolicy` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

    public function attachRolePolicy($roleName, $policyArn)
    {
        return $this->customWaiter(function () use ($roleName, $policyArn) {
            $this->iamClient->attachRolePolicy([
                'PolicyArn' => $policyArn,
```

```
                'RoleName' => $roleName,
            ]);
        });
    }
```

- API 詳細については、「 リファレンスAttachRolePolicy」の「」を参照してください。 AWS SDK for PHP API

`CreatePolicy`

次の例は、CreatePolicy を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

    public function createPolicy(string $policyName, string $policyDocument)
    {
        $result = $this->customWaiter(function () use ($policyName, $policyDocument)
 {
            return $this->iamClient->createPolicy([
                'PolicyName' => $policyName,
```

```
                'PolicyDocument' => $policyDocument,
            ]);
        });
        return $result['Policy'];
    }
```

- API 詳細については、「 リファレンスCreatePolicy」の「」を参照してください。 AWS SDK for PHP API

`CreateRole`

次の例は、CreateRole を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

    /**
     * @param string $roleName
     * @param string $rolePolicyDocument
     * @return array
```

```
     * @throws AwsException
     */
    public function createRole(string $roleName, string $rolePolicyDocument)
    {
        $result = $this->customWaiter(function () use ($roleName,
$rolePolicyDocument) {
            return $this->iamClient->createRole([
                'AssumeRolePolicyDocument' => $rolePolicyDocument,
                'RoleName' => $roleName,
            ]);
        });
        return $result['Role'];
    }
```

- API 詳細については、「 リファレンスCreateRole」の「」を参照してください。 AWS SDK for PHP API

`CreateServiceLinkedRole`

次の例は、`CreateServiceLinkedRole` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

    public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
$description = "")
    {
        $createServiceLinkedRoleArguments = ['AWSServiceName' => $awsServiceName];
        if ($customSuffix) {
            $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
        }
```

```
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
        }
        return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- API 詳細については、「 リファレンスCreateServiceLinkedRole」の「」を参照してください。AWS SDK for PHP API

`CreateUser`

次のコード例は、CreateUser を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";


    /**
     * @param string $name
     * @return array
     * @throws AwsException
     */
    public function createUser(string $name): array
    {
        $result = $this->iamClient->createUser([
            'UserName' => $name,
        ]);
```

```
        return $result['User'];
    }
```

- API 詳細については、「 リファレンスCreateUser」の「」を参照してください。 AWS SDK for PHP API

GetAccountPasswordPolicy

次の例は、GetAccountPasswordPolicy を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- API 詳細については、「 リファレンスGetAccountPasswordPolicy」の「」を参照してください。 AWS SDK for PHP API

GetPolicy

次のコード例は、GetPolicy を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
```

- API 詳細については、「 リファレンスGetPolicy」の「」を参照してください。 AWS SDK for PHP API

`GetRole`

次の例は、GetRole を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
```

```
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- API 詳細については、「 リファレンスGetRole」の「」を参照してください。 AWS SDK for PHP API

`ListAttachedRolePolicies`

次のコード例は、ListAttachedRolePolicies を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

    public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker =
 "", $maxItems = 0)
    {
        $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
        if ($pathPrefix) {
            $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
        }
        if ($marker) {
            $listAttachRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->iamClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
```

- API 詳細については、「 リファレンスListAttachedRolePolicies」の「」を参照してください。 AWS SDK for PHP API

`ListGroups`

次の例は、`ListGroups` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

    public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listGroupsArguments = [];
        if ($pathPrefix) {
            $listGroupsArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listGroupsArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listGroupsArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listGroups($listGroupsArguments);
    }
```

- API 詳細については、「 リファレンスListGroups」の「」を参照してください。 AWS SDK for PHP API

ListPolicies

次のコード例は、ListPolicies を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

    public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listPoliciesArguments = [];
        if ($pathPrefix) {
            $listPoliciesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listPoliciesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
```

- API 詳細については、「 リファレンスListPolicies」の「」を参照してください。 AWS SDK for PHP API

ListRolePolicies

次のコード例は、ListRolePolicies を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

    public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
    {
        $listRolePoliciesArguments = ['RoleName' => $roleName];
        if ($marker) {
            $listRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient->listRolePolicies($listRolePoliciesArguments);
        });
    }
```

- API 詳細については、「 リファレンスListRolePolicies」の「」を参照してください。 AWS SDK for PHP API

`ListRoles`

次のコード例は、`ListRoles` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

    /**
     * @param string $pathPrefix
     * @param string $marker
     * @param int $maxItems
     * @return Result
     * $roles = $service->listRoles();
     */
    public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listRolesArguments = [];
        if ($pathPrefix) {
            $listRolesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listRolesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listRolesArguments["MaxItems"] = $maxItems;
        }
        return $this->iamClient->listRoles($listRolesArguments);
    }
```

- API 詳細については、「 リファレンスListRoles」の「」を参照してください。 AWS SDK for PHP API

`ListSAMLProviders`

次の例は、`ListSAMLProviders` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- API 詳細については、「 リファレンス」の「ListSAMLProviders」を参照してください。 AWS SDK for PHP API

`ListUsers`

次のコード例は、`ListUsers` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$uuid = uniqid();
$service = new IAMService();

    public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listUsersArguments = [];
        if ($pathPrefix) {
            $listUsersArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listUsersArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listUsersArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listUsers($listUsersArguments);
```

```
    }
```

- API 詳細については、「 リファレンスListUsers」の「」を参照してください。 AWS SDK for PHP API

シナリオ

ユーザーを作成してロールを引き受ける

次のコードサンプルは、ユーザーを作成してロールを割り当てる方法を示しています。

> **Warning**
>
> セキュリティ上のリスクを回避するため、専用ソフトウェアの開発時や実際のデータの使用時には、認証にIAMユーザーを使用しないでください。代わりに、AWS IAM Identity Centerなどの ID プロバイダーとのフェデレーションを使用してください。

- 権限のないユーザーを作成します。
- 指定したアカウントに Amazon S3 バケットへのアクセス権限を付与するロールを作成します。
- ユーザーにロールを引き受けさせるポリシーを追加します。
- ロールを引き受け、一時的な認証情報を使用して S3 バケットを一覧表示しリソースをクリーンアップします。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
```

```
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("---------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("---------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
```

```
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
```

```
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- API 詳細については、「 AWS SDK for PHP APIリファレンス」の以下のトピックを参照してください。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

SDK で を使用する Kinesis の例 PHP

次のコード例は、Kinesis AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- サーバーレスサンプル

サーバーレスサンプル

Kinesis トリガーから Lambda 関数を呼び出す

次のコード例では、Kinesis ストリームからレコードを受信することによってトリガーされるイベントを受け取る、Lambda 関数の実装方法を示しています。この関数は Kinesis ペイロードを取得し、それを Base64 からデコードして、そのレコードの内容をログ記録します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda で Kinesis イベントを消費するPHP。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
```

```
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Kinesis トリガーを使用した Lambda 関数でのバッチアイテムの失敗のレポート

以下のコード例では、Kinesis ストリームからイベントを受け取る Lambda 関数のための、部分的なバッチレスポンスの実装方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda で Kinesis バッチアイテムの失敗をレポートするPHP。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php
```

```
# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
```

```
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

SDK で を使用する Lambda の例 PHP

次のコード例は、Lambda AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- アクション
- シナリオ
- サーバーレスサンプル

アクション

`CreateFunction`

次のコード例は、`CreateFunction` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    public function createFunction($functionName, $role, $bucketName, $handler)
    {
        //This assumes the Lambda function is in an S3 bucket.
        return $this->customWaiter(function () use ($functionName, $role,
$bucketName, $handler) {
            return $this->lambdaClient->createFunction([
                'Code' => [
                    'S3Bucket' => $bucketName,
                    'S3Key' => $functionName,
                ],
                'FunctionName' => $functionName,
                'Role' => $role['Arn'],
                'Runtime' => 'python3.9',
                'Handler' => "$handler.lambda_handler",
            ]);
        });
    }
```

- API 詳細については、「 リファレンスCreateFunction」の「」を参照してください。 AWS SDK for PHP API

`DeleteFunction`

次の例は、`DeleteFunction` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
public function deleteFunction($functionName)
{
    return $this->lambdaClient->deleteFunction([
        'FunctionName' => $functionName,
    ]);
}
```

- API 詳細については、「 リファレンスDeleteFunction」の「」を参照してください。 AWS SDK for PHP API

`GetFunction`

次の例は、`GetFunction` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
public function getFunction($functionName)
{
    return $this->lambdaClient->getFunction([
        'FunctionName' => $functionName,
    ]);
}
```

- API 詳細については、「 リファレンスGetFunction」の「」を参照してください。 AWS SDK for PHP API

`Invoke`

次の例は、`Invoke` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
public function invoke($functionName, $params, $logType = 'None')
{
    return $this->lambdaClient->invoke([
        'FunctionName' => $functionName,
        'Payload' => json_encode($params),
        'LogType' => $logType,
    ]);
}
```

- API 詳細については、「 AWS SDK for PHP APIリファレンス」の「呼び出し」を参照してください。

`ListFunctions`

次のコード例は、`ListFunctions` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    public function listFunctions($maxItems = 50, $marker = null)
    {
        if (is_null($marker)) {
            return $this->lambdaClient->listFunctions([
                'MaxItems' => $maxItems,
            ]);
        }

        return $this->lambdaClient->listFunctions([
            'Marker' => $marker,
            'MaxItems' => $maxItems,
        ]);
    }
```

- API 詳細については、「 リファレンスListFunctions」の「」を参照してください。 AWS SDK for PHP API

UpdateFunctionCode

次の例は、UpdateFunctionCode を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
    {
        return $this->lambdaClient->updateFunctionCode([
            'FunctionName' => $functionName,
            'S3Bucket' => $s3Bucket,
            'S3Key' => $s3Key,
        ]);
    }
```

- API 詳細については、「 リファレンスUpdateFunctionCode」の「」を参照してください。 AWS SDK for PHP API

`UpdateFunctionConfiguration`

次のコード例は、UpdateFunctionConfiguration を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
    public function updateFunctionConfiguration($functionName, $handler,
$environment = '')
    {
        return $this->lambdaClient->updateFunctionConfiguration([
            'FunctionName' => $functionName,
            'Handler' => "$handler.lambda_handler",
            'Environment' => $environment,
        ]);
    }
```

- API 詳細については、「 リファレンスUpdateFunctionConfiguration」の「」を参照してください。 AWS SDK for PHP API

シナリオ

サーバーレスアプリケーションを作成して写真の管理

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

PHP に関する SDK

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

関数の使用を開始します

次のコードサンプルは、以下の操作方法を示しています。

- IAM ロールと Lambda 関数を作成し、ハンドラーコードをアップロードします。
- 1 つのパラメーターで関数を呼び出して、結果を取得します。
- 関数コードを更新し、環境変数で設定します。
- 新しいパラメーターで関数を呼び出して、結果を取得します。返された実行ログを表示します。
- アカウントの関数を一覧表示し、リソースをクリーンアップします。

詳細については、「コンソールで Lambda 関数を作成する」を参照してください。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
namespace Lambda;

use Aws\S3\S3Client;
use GuzzleHttp\Psr7\Stream;
```

```
use Iam\IAMService;

class GettingStartedWithLambda
{
    public function run()
    {
        echo("\n");
        echo("---------------------------------------\n");
        print("Welcome to the AWS Lambda getting started demo using PHP!\n");
        echo("---------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $iamService = new IAMService();
        $s3client = new S3Client($clientArgs);
        $lambdaService = new LambdaService();

        echo "First, let's create a role to run our Lambda code.\n";
        $roleName = "test-lambda-role-$uniqid";
        $rolePolicyDocument = "{
            \"Version\": \"2012-10-17\",
            \"Statement\": [
                {
                    \"Effect\": \"Allow\",
                    \"Principal\": {
                        \"Service\": \"lambda.amazonaws.com\"
                    },
                    \"Action\": \"sts:AssumeRole\"
                }
            ]
        }";
        $role = $iamService->createRole($roleName, $rolePolicyDocument);
        echo "Created role {$role['RoleName']}.\n";

        $iamService->attachRolePolicy(
            $role['RoleName'],
            "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
        );
        echo "Attached the AWSLambdaBasicExecutionRole to {$role['RoleName']}.\n";
```

```
        echo "\nNow let's create an S3 bucket and upload our Lambda code there.\n";
        $bucketName = "test-example-bucket-$uniqid";
        $s3client->createBucket([
            'Bucket' => $bucketName,
        ]);
        echo "Created bucket $bucketName.\n";

        $functionName = "doc_example_lambda_$uniqid";
        $codeBasic = __DIR__ . "/lambda_handler_basic.zip";
        $handler = "lambda_handler_basic";
        $file = file_get_contents($codeBasic);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "Uploaded the Lambda code.\n";

        $createLambdaFunction = $lambdaService->createFunction($functionName, $role,
 $bucketName, $handler);
        // Wait until the function has finished being created.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['State'] == "Pending");
        echo "Created Lambda function {$getLambdaFunction['Configuration']
['FunctionName']}.\n";

        sleep(1);

        echo "\nOk, let's invoke that Lambda code.\n";
        $basicParams = [
            'action' => 'increment',
            'number' => 3,
        ];
        /** @var Stream $invokeFunction */
        $invokeFunction = $lambdaService->invoke($functionName, $basicParams)
['Payload'];
        $result = json_decode($invokeFunction->getContents())->result;
        echo "After invoking the Lambda code with the input of
 {$basicParams['number']} we received $result.\n";

        echo "\nSince that's working, let's update the Lambda code.\n";
```

```
        $codeCalculator = "lambda_handler_calculator.zip";
        $handlerCalculator = "lambda_handler_calculator";
        echo "First, put the new code into the S3 bucket.\n";
        $file = file_get_contents($codeCalculator);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "New code uploaded.\n";

        $lambdaService->updateFunctionCode($functionName, $bucketName,
 $functionName);
        // Wait for the Lambda code to finish updating.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
 "Successful");
        echo "New Lambda code uploaded.\n";

        $environment = [
            'Variable' => ['Variables' => ['LOG_LEVEL' => 'DEBUG']],
        ];
        $lambdaService->updateFunctionConfiguration($functionName,
 $handlerCalculator, $environment);
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
 "Successful");
        echo "Lambda code updated with new handler and a LOG_LEVEL of DEBUG for more
 information.\n";

        echo "Invoke the new code with some new data.\n";
        $calculatorParams = [
            'action' => 'plus',
            'x' => 5,
            'y' => 4,
        ];
        $invokeFunction = $lambdaService->invoke($functionName, $calculatorParams,
 "Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
```

```
        echo "Indeed, {$calculatorParams['x']} + {$calculatorParams['y']} does equal
$result.\n";
        echo "Here's the extra debug info: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nBut what happens if you try to divide by zero?\n";
        $divZeroParams = [
            'action' => 'divide',
            'x' => 5,
            'y' => 0,
        ];
        $invokeFunction = $lambdaService->invoke($functionName, $divZeroParams,
"Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
        echo "You get a |$result| result.\n";
        echo "And an error message: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nHere's all the Lambda functions you have in this Region:\n";
        $listLambdaFunctions = $lambdaService->listFunctions(5);
        $allLambdaFunctions = $listLambdaFunctions['Functions'];
        $next = $listLambdaFunctions->get('NextMarker');
        while ($next != false) {
            $listLambdaFunctions = $lambdaService->listFunctions(5, $next);
            $next = $listLambdaFunctions->get('NextMarker');
            $allLambdaFunctions = array_merge($allLambdaFunctions,
$listLambdaFunctions['Functions']);
        }
        foreach ($allLambdaFunctions as $function) {
            echo "{$function['FunctionName']}\n";
        }

        echo "\n\nAnd don't forget to clean up your data!\n";

        $lambdaService->deleteFunction($functionName);
        echo "Deleted Lambda function.\n";
        $iamService->deleteRole($role['RoleName']);
        echo "Deleted Role.\n";
        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
```

```
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Deleted all objects from the S3 bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);
        echo "Deleted the bucket.\n";
    }
}
```

- API 詳細については、「 AWS SDK for PHP APIリファレンス」の以下のトピックを参照してください。
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

サーバーレスサンプル

Lambda 関数の Amazon RDS データベースへの接続

次のコード例は、 RDS データベースに接続する Lambda 関数を実装する方法を示しています。この関数は、シンプルなデータベースリクエストを実行し、結果を返します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda 関数の Amazon RDS データベースに接続するPHP。

```
<?php
```

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
```

```
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
```

```
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Kinesis トリガーから Lambda 関数を呼び出す

次のコード例では、Kinesis ストリームからレコードを受信することによってトリガーされるイベントを受け取る、Lambda 関数の実装方法を示しています。この関数は Kinesis ペイロードを取得し、それを Base64 からデコードして、そのレコードの内容をログ記録します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda で Kinesis イベントを消費するPHP。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
```

```
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

DynamoDB トリガーから Lambda 関数を呼び出す

次のコード例は、DynamoDB ストリームからレコードを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示しています。関数は DynamoDB ペイロードを取得し、レコードの内容を口グ記録します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda で DynamoDB イベントを消費するPHP。

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data
```

```
            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Amazon DocumentDB トリガーから Lambda 関数を呼び出す

次のコード例は、DocumentDB 変更ストリームからレコードを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示しています。関数は DocumentDB ペイロードを取得し、レコードの内容をログ記録します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda で Amazon DocumentDB イベントを消費するPHP。

```
<?php

require __DIR__.'/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
```

```
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Amazon S3 トリガーから Lambda 関数を呼び出す

次のコード例は、S3 バケットにオブジェクトをアップロードすることによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数は、イベントパラメータから S3 バケット名とオブジェクトキーを取得し、Amazon S3 を呼び出しAPIてオブジェクトのコンテンツタイプを取得してログに記録します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda で S3 イベントを消費するPHP。

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
```

```
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Amazon SNSトリガーから Lambda 関数を呼び出す

次のコード例は、 SNSトピックからメッセージを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示しています。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda でSNSイベントを消費するPHP。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';
```

```
use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Amazon SQSトリガーから Lambda 関数を呼び出す

次のコード例は、キューからメッセージを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示していますSQS。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda でSQSイベントを消費するPHP。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Kinesis トリガーを使用した Lambda 関数でのバッチアイテムの失敗のレポート

以下のコード例では、Kinesis ストリームからイベントを受け取る Lambda 関数のための、部分的なバッチレスポンスの実装方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda で Kinesis バッチアイテムの失敗をレポートするPHP。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
```

```
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

DynamoDB トリガーで Lambda 関数のバッチアイテムの失敗をレポートする

次のコード例は、DynamoDB ストリームからイベントを受信する Lambda 関数に部分的なバッチレスポンスを実装する方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用した Lambda での DynamoDB バッチアイテムの失敗のレポートPHP。

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
```

```
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Amazon SQSトリガーを使用した Lambda 関数のバッチアイテム失敗のレポート

次のコード例は、 SQSキューからイベントを受信する Lambda 関数に部分的なバッチレスポンスを実装する方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda でSQSバッチアイテムの失敗をレポートするPHP。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
```

```
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

SDK に を使用する Amazon RDSの例 PHP

次のコード例は、Amazon AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示していますRDS。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

アクション

`CreateDBInstance`

次の例は、`CreateDBInstance` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require __DIR__ . '/vendor/autoload.php';
```

```
use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$dbClass = 'db.t2.micro';
$storage = 5;
$engine = 'MySQL';
$username = 'MyUser';
$password = 'MyPassword';

try {
    $result = $rdsClient->createDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBInstanceClass' => $dbClass,
        'AllocatedStorage' => $storage,
        'Engine' => $engine,
        'MasterUsername' => $username,
        'MasterUserPassword' => $password,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API 詳細については、「 リファレンス」の「CreateDBInstance」を参照してください。 AWS SDK for PHP API

`CreateDBSnapshot`

次のコード例は、CreateDBSnapshot を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$snapshotName = '<<{{backup_2018_12_25}}>>';

try {
    $result = $rdsClient->createDBSnapshot([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBSnapshotIdentifier' => $snapshotName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API 詳細については、「 リファレンス」の「CreateDBSnapshot」を参照してください。 AWS SDK for PHP API

`DeleteDBInstance`

次のコード例は、`DeleteDBInstance` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-1'
]);

$dbIdentifier = '<<{{db-identifier}}>>';

try {
    $result = $rdsClient->deleteDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API 詳細については、「 リファレンス」の「DeleteDBInstance」を参照してください。 AWS SDK for PHP API

`DescribeDBInstances`

次の例は、DescribeDBInstances を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

try {
    $result = $rdsClient->describeDBInstances();
    foreach ($result['DBInstances'] as $instance) {
        print('<p>DB Identifier: ' . $instance['DBInstanceIdentifier']);
        print('<br />Endpoint: ' . $instance['Endpoint']["Address"]
            . ':' . $instance['Endpoint']["Port"]);
        print('<br />Current Status: ' . $instance["DBInstanceStatus"]);
        print('</p>');
    }
    print(" Raw Result ");
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API 詳細については、「 リファレンス」の「DescribeDBInstances」を参照してください。 AWS SDK for PHP API

シナリオ

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信するウェブアプリケーションを作成する方法を示しています。

PHP に関する SDK

を使用して、Amazon RDS データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon) を使用してレポートを E メールで送信するウェブアプリケーション AWS SDK for PHP を作成する方法を示しますSES。この例では、React.js で構築されたフロントエンドを使用してRESTfulPHPバックエンドとやり取りします。

- React.js ウェブアプリケーションを AWS サービスと統合します。
- Amazon RDSテーブル内の項目を一覧表示、追加、更新、削除します。
- Amazon を使用して、フィルタリングされた作業項目の E メールレポートを送信しますSES。
- 付属の AWS CloudFormation スクリプトを使用してサンプルリソースをデプロイおよび管理します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

サーバーレスサンプル

Lambda 関数の Amazon RDS データベースへの接続

次のコード例は、 RDS データベースに接続する Lambda 関数を実装する方法を示しています。この関数は、シンプルなデータベースリクエストを実行し、結果を返します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda 関数の Amazon RDS データベースに接続するPHP。

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
```

```
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
```

```
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

SDK の を使用した Amazon RDS Data Service の例 PHP

次のコード例は、Amazon RDS Data Service AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- シナリオ

シナリオ

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信するウェブアプリケーションを作成する方法を示しています。

PHP に関する SDK

を使用して、Amazon RDS データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon) を使用してレポートを E メールで送信するウェブアプリケーション AWS SDK for PHP を作成する方法を示しますSES。この例では、React.js で構築されたフロントエンドを使用してRESTfulPHPバックエンドとやり取りします。

- React.js ウェブアプリケーションを AWS サービスと統合します。
- Amazon RDSテーブル内の項目を一覧表示、追加、更新、削除します。
- Amazon を使用して、フィルタリングされた作業項目の E メールレポートを送信しますSES。
- 付属の AWS CloudFormation スクリプトを使用してサンプルリソースをデプロイおよび管理します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

SDK に を使用する Amazon Rekognition の例 PHP

次のコード例は、Amazon Rekognition AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。 Amazon Rekognition

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- シナリオ

シナリオ

サーバーレスアプリケーションを作成して写真の管理

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

PHP に関する SDK

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

で を使用する Amazon S3 SDK の例 PHP

次のコード例は、Amazon S3 AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

「基本」は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

開始方法

Hello Amazon S3

次のコード例は、Amazon S3 の使用を開始する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
use Aws\S3\S3Client;

$client = new S3Client(['region' => 'us-west-2']);
$results = $client->listBuckets();
var_dump($results);
```

- API 詳細については、「 リファレンスListBuckets」の「」を参照してください。 AWS SDK for PHP API

トピック

- 基礎
- アクション
- シナリオ
- サーバーレスサンプル

基礎

基本を学ぶ

次のコードサンプルは、以下の操作方法を示しています。

- バケットを作成し、そこにファイルをアップロードします。
- バケットからオブジェクトをダウンロードします。
- バケット内のサブフォルダにオブジェクトをコピーします。
- バケット内のオブジェクトを一覧表示します。
- バケットオブジェクトとバケットを削除します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
        echo("\n");
        echo("---------------------------------------\n");
        print("Welcome to the Amazon S3 getting started demo using PHP!\n");
        echo("---------------------------------------\n");

        $region = 'us-west-2';

        $this->s3client = new S3Client([
                'region' => $region,
        ]);
        /* Inline declaration example
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
        */

        $this->bucketName = "doc-example-bucket-" . uniqid();

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
```

```
        }

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception->getMessage();
            exit("Please fix error with object copying before continuing.");
        }

        try {
```

```
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (count($check) <= 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
```

```
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }

        echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- API 詳細については、「 AWS SDK for PHP APIリファレンス」の以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

アクション

`CopyObject`

次の例は、`CopyObject` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

オブジェクトの単純なコピー。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
```

```
        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception->getMessage();
            exit("Please fix error with object copying before continuing.");
        }
```

- API 詳細については、「 リファレンスCopyObject」の「」を参照してください。 AWS SDK for PHP API

`CreateBucket`

次の例は、CreateBucket を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

バケットを作成します。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
```

```
            exit("Please fix error with bucket creation before continuing.");
        }
```

- API 詳細については、「 リファレンスCreateBucket」の「」を参照してください。 AWS SDK for PHP API

`DeleteBucket`

次のコード例は、DeleteBucket を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

空のバケットを削除します。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

- API 詳細については、「 リファレンスDeleteBucket」の「」を参照してください。 AWS SDK for PHP API

`DeleteObjects`

次の例は、DeleteObjects を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

キーのリストからオブジェクトのセットを削除します。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (count($check) <= 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

- API 詳細については、「 リファレンスDeleteObjects」の「」を参照してください。 AWS SDK for PHP API

GetObject

次のコード例は、GetObject を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

オブジェクトを取得します。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }
```

- API 詳細については、「 リファレンスGetObject」の「」を参照してください。 AWS SDK for PHP API

ListObjectsV2

次のコード例は、ListObjectsV2 を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

バケット内のオブジェクトを一覧表示します。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }
```

- API 詳細については、「 リファレンス」のListObjectsV2」を参照してください。 AWS SDK for PHP API

`PutObject`

次のコード例は、`PutObject` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

オブジェクトをバケットにアップロードします。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }
```

- API 詳細については、「 リファレンスPutObject」の「」を参照してください。 AWS SDK for PHP API

シナリオ

署名付き を作成する URL

次のコード例は、Amazon S3 URLの署名付き を作成し、オブジェクトをアップロードする方法を示しています。 Amazon S3

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
```

```
use DateTime;

require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the S3
 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in the
 given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
 next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
```

```
$runner->run();
```

サーバーレスアプリケーションを作成して写真の管理

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

PHP に関する SDK

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

サーバーレスサンプル

Amazon S3 トリガーから Lambda 関数を呼び出す

次のコード例は、S3 バケットにオブジェクトをアップロードすることによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数は、イベントパラメータから S3 バケット名とオブジェクトキーを取得し、Amazon S3 を呼び出しAPIてオブジェクトのコンテンツタイプを取得してログに記録します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda で S3 イベントを消費するPHP。

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
```

```
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

SDK で を使用する Amazon SESの例 PHP

次のコード例は、Amazon AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示していますSES。

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- シナリオ

シナリオ

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信するウェブアプリケーションを作成する方法を示しています。

PHP に関する SDK

を使用して、Amazon RDS データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon) を使用してレポートを E メールで送信するウェブアプリケーション AWS SDK for PHP を作成する方法を示しますSES。この例では、React.js で構築されたフロントエンドを使用してRESTfulPHPバックエンドとやり取りします。

- React.js ウェブアプリケーションを AWS サービスと統合します。
- Amazon RDSテーブル内の項目を一覧表示、追加、更新、削除します。
- Amazon を使用して、フィルタリングされた作業項目の E メールレポートを送信しますSES。
- 付属の AWS CloudFormation スクリプトを使用してサンプルリソースをデプロイおよび管理します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

SDK に を使用する Amazon SNSの例 PHP

次のコード例は、Amazon AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示していますSNS。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS サービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- アクション
- シナリオ
- サーバーレスサンプル

アクション

`CheckIfPhoneNumberIsOptedOut`

次の例は、`CheckIfPhoneNumberIsOptedOut` を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
```

```
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API 詳細については、「 リファレンスCheckIfPhoneNumberIsOptedOut」の「」を参照してください。 AWS SDK for PHP API

`ConfirmSubscription`

次の例は、ConfirmSubscription を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */
```

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 詳細については、「 リファレンスConfirmSubscription」の「」を参照してください。 AWS SDK for PHP API

`CreateTopic`

次のコード例は、`CreateTopic` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;


/**
 * Create a Simple Notification Service topics in your AWS account at the requested
 region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API 詳細については、「 リファレンスCreateTopic」の「」を参照してください。 AWS SDK for PHP API

DeleteTopic

次の例は、DeleteTopic を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 詳細については、「 リファレンスDeleteTopic」の「」を参照してください。 AWS SDK for PHP API

GetSMSAttributes

次の例は、GetSMSAttributes を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API 詳細については、「 リファレンス」の「GetSMSAttributes」を参照してください。 AWS SDK for PHP API

`GetTopicAttributes`

次の例は、GetTopicAttributes を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 詳細については、「 リファレンスGetTopicAttributes」の「」を参照してください。 AWS SDK for PHP API

`ListPhoneNumbersOptedOut`

次のコード例は、`ListPhoneNumbersOptedOut` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages from
 your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API 詳細については、「 リファレンスListPhoneNumbersOptedOut」の「」を参照してください。 AWS SDK for PHP API

`ListSubscriptions`

次のコード例は、`ListSubscriptions` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 詳細については、「 リファレンスListSubscriptions」の「」を参照してください。 AWS SDK for PHP API

`ListTopics`

次のコード例は、`ListTopics` を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of the requester's topics from your AWS SNS account in the region
 specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 詳細については、「 リファレンスListTopics」の「」を参照してください。 AWS SDK for PHP API

Publish

次の例は、Publish を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
```

```
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API 詳細については、「 リファレンス」の「パブリッシュ」を参照してください。 AWS SDK for PHP API

SetSMSAttributes

次のコード例は、SetSMSAttributes を使用する方法を示しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API 詳細については、「 リファレンス」の「SetSMSAttributes」を参照してください。 AWS SDK for PHP API

SetTopicAttributes

次の例は、SetTopicAttributes を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
```

```
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 詳細については、「 リファレンスSetTopicAttributes」の「」を参照してください。 AWS SDK for PHP API

`Subscribe`

次の例は、Subscribe を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

E メールアドレスをトピックにサブスクライブします。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

HTTP エンドポイントをトピックにサブスクライブします。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 詳細については、「 AWS SDK for PHP APIリファレンス」の「サブスクライブ」を参照してください。

Unsubscribe

次の例は、Unsubscribe を使用する方法を説明しています。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。

- API 詳細については、「 AWS SDK for PHP APIリファレンス」の「サブスクリプション解除」を参照してください。

シナリオ

サーバーレスアプリケーションを作成して写真の管理

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

PHP に関する SDK

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

SMS テキストメッセージを発行する

次のコード例は、Amazon を使用してSMSメッセージを発行する方法を示していますSNS。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。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。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API 詳細については、「 リファレンス」の「パブリッシュ」を参照してください。 AWS SDK for PHP API

サーバーレスサンプル

Amazon SNSトリガーから Lambda 関数を呼び出す

次のコード例は、 SNSトピックからメッセージを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示しています。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda でSNSイベントを消費するPHP。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
```

```
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

SDK に を使用する Amazon SQSの例 PHP

次のコード例は、Amazon AWS SDK for PHP で を使用してアクションを実行し、一般的なシナリオを実装する方法を示していますSQS。

各例には、完全なソースコードへのリンクが含まれています。このリンクには、コンテキスト内でコードをセットアップして実行する方法の手順が記載されています。

トピック

- サーバーレスサンプル

サーバーレスサンプル

Amazon SQSトリガーから Lambda 関数を呼び出す

次のコード例は、キューからメッセージを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示していますSQS。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda でSQSイベントを消費するPHP。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}
```

```
$logger = new StderrLogger();
return new Handler($logger);
```

Amazon SQSトリガーを使用した Lambda 関数のバッチアイテム失敗のレポート

次のコード例は、 SQSキューからイベントを受信する Lambda 関数に部分的なバッチレスポンスを実装する方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

PHP に関する SDK

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda でSQSバッチアイテムの失敗をレポートするPHP。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
```

```
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

のセキュリティ AWS SDK for PHP

クラウドセキュリティは Amazon Web Services (AWS) の最優先事項です。 AWS のお客様は、セキュリティを非常に重視する組織の要件を満たせるように構築されたデータセンターとネットワークアーキテクチャーから利点を得ます。セキュリティは、 AWS とユーザーの間で共有される責任です。責任共有モデルでは、これを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しています。

クラウドのセキュリティ — AWS クラウドで提供されるすべてのサービスを実行するインフラストラクチャ AWS を保護し、安全に使用できるサービスを提供します。当社のセキュリティ責任は で最優先事項であり AWS、当社のセキュリティの有効性は、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者によって定期的にテストおよび検証されています。

クラウドにおけるセキュリティ — お客様の責任は、使用している AWS サービス、およびデータの機密性、組織の要件、適用可能な法律や規制などのその他の要因によって決まります。

トピック

- AWS SDK for PHPでのデータ保護
- Identity and Access Management
- この AWS 製品またはサービスのコンプライアンス検証
- この AWS 製品またはサービスの耐障害性
- この AWS 製品またはサービスのインフラストラクチャセキュリティ
- Amazon S3 暗号化クライアントの移行

AWS SDK for PHPでのデータ保護

責任 AWS 共有モデル、 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 はすべての 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責任があります AWS クラウド。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 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FAQ」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」のAWS 「責任共有モデル」とGDPRブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management () を使用して個々のユーザーを設定することをお勧めしますIAM。

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。1TLS.2 が必要で、1.3 TLS をお勧めします。
- を使用して APIとユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS サービス。
- 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 AWS を介して にアクセスするときに FIPS 140-3 検証済みの暗号化モジュールが必要な場合はAPI、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) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、、 AWS SDK for PHP または を使用して または他の AWS サービス を操作する場合API AWS CLIも同様です AWS SDKs。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。URL を外部サーバーに提供する場合は、そのサーバーへのリクエストを検証URLするために認証情報を に含めないことを強くお勧めします。

Identity and Access Management

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 AWS サービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は追加料金なしで AWS サービス 使用できる です。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- の AWS サービス 仕組み IAM
- AWS ID 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) の使用方法は、 で行う作業によって異なります AWS。

サービスユーザー – AWS サービス を使用してジョブを実行する場合、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。さらに多くの AWS 機能を使用して作業を行う場合は、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しておくと、管理者に適切な許可をリクエストするうえで役立ちます。の機能にアクセスできない場合は AWS、AWS ID とアクセスのトラブルシューティング「」または AWS サービス 使用している のユーザーガイドを参照してください。

サービス管理者 – 社内の AWS リソースを担当している場合は、通常、 へのフルアクセスがあります AWS。サービスユーザーがどの AWS 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。次に、サービスユーザーのアクセス許可を変更するリクエストをIAM管理者に送信する必要があります。このページの情報を確認して、 の基本概念を理解してくださいIAM。会社IAMで を使用する方法の詳細については AWS、使用している の AWS サービス ユーザーガイドを参照してください。

IAM 管理者 – IAM管理者の場合は、 へのアクセスを管理するポリシーの作成方法の詳細について確認する場合があります AWS。で使用できる AWS アイデンティティベースのポリシーの例を表示するにはIAM、 AWS サービス 使用している のユーザーガイドを参照してください。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAMユーザーとして AWS アカウントのルートユーザー、または IAMロールを引き受けることによって認証 （ にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center （ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID としてサインインすると、管理者は以前に IAMロールを使用して ID フェデレーションをセットアップしていました。フェデレーション AWS を使用して にアクセスすると、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。へのサインインの詳細については AWS、「 ユーザーガイド」の「 へのサインイン AWS アカウント方法AWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 はソフトウェア開発キット (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、 認証情報を使用して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。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、「 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を高めるために多要素認証 (MFA) を使用することをお勧めします。詳細については、「 AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「 ユーザーガイド」の「 での多要素認証 (MFA) AWS IAM の使用」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての AWS サービス およびリソースへの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 E メールアドレスとパスワードでサインインすることでアクセスできます。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「ルートユーザーの認証情報を必要とするタスク」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、一時的な認証情報を使用して にアクセスするための ID プロバイダーとのフェデレーションの使用を要求 AWS サービス します。

フェデレーティッド ID 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、または ID ソースを通じて提供された認証情報 AWS サービス を使用して にアクセスするユーザーです。フェデレーティッド ID が にアクセスすると AWS アカウント、ロールが引き受けられ、ロールは一時的な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 Identity Center でユーザーとグループを作成することも、独自の ID ソース内のユーザーとグループのセットに接続して同期して、すべての AWS アカウント とアプリケーションで使用することもできます。IAM Identity Center の詳細については、「 ユーザーガイド」のIAM「Identity Center とはAWS IAM Identity Center 」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、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 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持つIAMユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAMユーザーとの長期的な認証情報を必要とする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、「 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でアクセスキーを定期的にローテーションするIAM」を参照してください。

IAM グループは、IAMユーザーのコレクション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、 という名前のグループを作成しIAMAdmins、そのグループにIAMリソースを管理するアクセス許可を付与できます。

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、ロールでは一時認証情報が提供されます。詳細については、「 ユーザーガイド」のIAM「 (ロールではなく) ユーザーを作成する場合IAM」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーと似ていますが、特定のユーザーに関連付けられていません。IAM ロール を切り替える AWS Management Console ことで、 で 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、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタム を使用しますURL。ロールの使用方法の詳細については、「 ユーザーガイド」のIAM「ロールの使用IAM」を参照してください。

IAM 一時的な認証情報を持つ ロールは、以下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 ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールの詳細については、「 ユーザーガイド」の「サードパーティー ID プロバイダーのロールの作成IAM」を参照してください。IAM Identity Center を使用する場合は、アクセス許可セットを設定します。ID が認証後にアクセスできる内容を制御するために、IAMIdentity Center はアクセス

許可セットを のロールに関連付けますIAM。アクセス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「アクセス許可セット」を参照してください。

- 一時的なIAMユーザーアクセス許可 – IAM ユーザーまたはロールは、 IAMロールを引き受けて、特定のタスクに対して異なるアクセス許可を一時的に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、別のアカウントのユーザー (信頼されたプリンシパル) がアカウントのリソースにアクセスすることを許可できます。クロスアカウントアクセスを許可する主な方法は、ロールを使用することです。ただし、一部の では AWS サービス、(ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスのロールとリソースベースのポリシーの違いについては、「 ユーザーガイド」の「 でのクロスアカウントリソースアクセスIAMIAM」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS サービス を使用します AWS サービス。例えば、サービスで呼び出しを行うと、そのサービスが Amazon でアプリケーションを実行EC2したり、Amazon S3 にオブジェクトを保存したりするのが一般的です。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンク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 でアクションを実行すると AWS、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、 を呼び出すプリンシパルのアクセス許可を AWS サービス、ダウンストリームサービス AWS サービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS サービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するための権限が必要です。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 – サービスロールは、ユーザーに代わってアクションを実行するためにサービスが引き受ける IAMロールです。IAM 管理者は、 内からサービスロールを作成、変更、削除できますIAM。詳細については、「 ユーザーガイド」の「 にアクセス許可を委任するロールの作成 AWS サービスIAM」を参照してください。
  - サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS サービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、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アクセス許可を表示できますが、編集することはできません。
- Amazon で実行されているアプリケーション EC2 – IAMロールを使用して、EC2インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管

理できます。これは、EC2インスタンス内にアクセスキーを保存するよりも望ましいです。 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、インスタンスにアタッチ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 ロールが含まれており、EC2インスタンスで実行されているプログラムが一時的な認証情報を取得できるようにします。詳細については、「 ユーザーガイド」の「 IAMロールを使用して Amazon EC2インスタンスで実行されているアプリケーションにアクセス許可を付与するIAM」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、「 ユーザーガイド」の「 (ユーザーではなく) IAMロールを作成する場合IAM」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 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) AWS がリクエストを行うときに、これらのポリシー AWS 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。ほとんどのポリシーはJSONドキュメント AWS として 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、「 ユーザーガイド」のJSON「ポリシーの概要IAM」を参照してください。

管理者はポリシーを使用して AWS JSON、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対してどのような条件下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。必要なリソースに対してアクションを実行するアクセス許可をユーザーに付与するために、IAM管理者はIAMポリシーを作成できます。その後、管理者はIAMポリシーをロールに追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行に使用するメソッドに関係なく、アクションのアクセス許可を定義します。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。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS からロール情報を取得できますAPI。

アイデンティティベースのポリシー

ID ベースのポリシーは、IAMユーザー、ユーザーのグループ、ロールなどの ID にアタッチできるJSONアクセス許可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、「 ユーザーガイド」のIAM「ポリシーの作成IAM」を参照してください。

アイデンティティベースのポリシーは、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ています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。管理ポリシーとインラインポリシーのどちらかを選択する方法については、IAM ユーザーガイドの「管理ポリシーとインラインポリシーの選択」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、リソースにアタッチする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の例としては、IAMロールの信頼ポリシー や Amazon S3 バケットポリシー などがあります。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コントロールできます。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、フェデレーティッドユーザー、または を含めることができます AWS サービス。

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IAMでは、 の AWS 管理ポリシーを使用できません。

アクセスコントロールリスト (ACLs)

アクセスコントロールリスト (ACLs) は、リソースへのアクセス許可を持つプリンシパル (アカウントメンバー、ユーザー、またはロール) を制御します。ACLs 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を使用しません。

Amazon S3、 AWS WAF、および Amazon VPCは、 をサポートするサービスの例ですACLs。の詳細についてはACLs、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 は、一般的ではない追加のポリシータイプをサポートします。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がIAMエンティティ (IAMユーザーまたはロール) に付与できるアクセス許可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 (SCPs) – SCPsは、 の組織または組織単位 (OU) に対する最大アクセス許可を指定するJSONポリシーです AWS Organizations。 AWS Organizations は、 AWS アカウント ビジネスが所有する複数の をグループ化して一元管理するためのサービスです。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 (SCPs) をアカウントの一部またはすべてに適用できます。は、各 を含むメンバーアカウントのエンティティのアクセス許可SCP を制限します AWS アカウントのルートユーザー。Organizations と の詳細についてはSCPs、「 AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」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、「 ユーザーガイド」の「セッションポリシーIAM」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係する場合にリクエストを許可するかどうか AWS を決定する方法については、「 ユーザーガイド」の「ポリシー評価ロジックIAM」を参照してください。

の AWS サービス 仕組み IAM

がほとんどの IAM 機能と AWS サービス 連携する方法の概要を把握するには、 IAMユーザーガイドのAWS 「 と連携する のサービスIAM」を参照してください。

で特定の を使用する方法についてはIAM、関連する AWS サービス サービスのユーザーガイドのセキュリティセクションを参照してください。

AWS ID とアクセスのトラブルシューティング

次の情報は、 および の使用時に発生する可能性がある一般的な問題の診断 AWS と修正に役立ちますIAM。

トピック

- でアクションを実行する権限がない AWS
- iam を実行する権限がありません。PassRole
- 自分の 以外のユーザーに自分の AWS リソース AWS アカウント へのアクセスを許可したい

でアクションを実行する権限がない AWS

「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。

次の例のエラーは、mateojacksonIAMユーザーが コンソールを使用して架空の*my-example-widget*リソースの詳細を表示しようとしているが、架空のawes:*GetWidget*アクセス許可がない場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

この場合、awes:*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 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam を実行する権限がありません。PassRole

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 AWS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 サービス では、新しい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作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

次の例のエラーは、 というIAMユーザーがコンソールを使用して marymajor でアクションを実行しようする場合に発生します AWS。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。メアリーには、ロールをサービスに渡す許可がありません。

```
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 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

自分の 以外のユーザーに自分の AWS リソース AWS アカウント 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 (ACLs) をサポートするサービスの場合、これらのポリシーを使用して、ユーザーにリソースへのアクセスを許可できます。

詳細については、以下を参照してください。

- がこれらの機能 AWS をサポートしているかどうかを確認するには、「」を参照してくださいの AWS サービス 仕組み IAM。
- 所有している のリソースへのアクセスを提供する方法については、 AWS アカウント 「 ユーザーガイド」の「所有 AWS アカウント している別の のIAMユーザーへのアクセスを提供するIAM」を参照してください。
- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、 IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを通じ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「外部認証されたユーザーへのアクセスの提供 (ID フェデレーション) 」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については、ユーザーガイドの「 でのクロスアカウントリソースアクセスIAMIAM」を参照してください。

この AWS 製品またはサービスのコンプライアンス検証

AWS サービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、コンプライアンスプログラムAWS サービス による対象範囲内のコンプライアンスプログラムを参照し、関心のあるコンプライアンスプログラムを選択します。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。

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 AWS Artifact。詳細については、「 でのレポートのダウンロード AWS Artifact」の」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS サービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ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 AWS を にデプロイする手順について説明します。
- アマゾン ウェブ サービスHIPAAのセキュリティとコンプライアンスのためのアーキテクチャ – このホワイトペーパーでは、企業が AWS を使用して HIPAA対象アプリケーションを作成する方法について説明します。

> **Note**
>
> すべての AWS サービス がHIPAA対象となるわけではありません。詳細については、HIPAA「対象サービスリファレンス」を参照してください。

- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド – コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS サービス 、複数のフレームワーク (米国国立標準技術研究所 (NIST) 、Payment Card Industry Security Standards Council ()、PCI国際標準化機構 (ISO) など) のセキュリティコントロールにマッピングするためのベストプラクティスをまとめています。
- 「 デベロッパーガイド」の「 ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config

- AWS Security Hub – これにより AWS サービス 、 内のセキュリティ状態を包括的に確認できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、Security Hub のコントロールリファレンスを参照してください。
- Amazon GuardDuty – これにより AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS サービス を検出します。 GuardDuty は、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことでDSS、 PCI などのさまざまなコンプライアンス要件に対応するのに役立ちます。
- AWS Audit Manager – これにより AWS サービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて責任共有モデルに従います。 AWS サービスセキュリティ情報については、AWS 「サービスセキュリティドキュメント」ページとAWS 、 AWS コンプライアンスプログラム によるコンプライアンスの取り組みの対象となるサービスを参照してください。

この AWS 製品またはサービスの耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。

AWS リージョン は、低レイテンシー、高スループット、および高度に冗長なネットワークで接続された、物理的に分離および隔離された複数のアベイラビリティーゾーンを提供します。

アベイラビリティーゾーンでは、ゾーン間で中断することなく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が高く、フォールトトレラントで、スケーラブルです。

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。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて責任共有モデルに従います。 AWS サービスセキュリティ情報については、AWS 「サービスセキュリティドキュメント」ページとAWS 、 AWS コンプライアンスプログラム によるコンプライアンスの取り組みの対象となるサービスを参照してください。

この AWS 製品またはサービスのインフラストラクチャセキュリティ

この AWS 製品またはサービスはマネージドサービスを使用するため、 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 。 AWS セキュリティサービスと がインフラストラクチャ AWS 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、「 Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

が AWS 公開したAPI呼び出しを使用して、ネットワーク経由でこの AWS 製品またはサービスにアクセスします。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) 。1TLS.2 が必要で、1.3 TLS をお勧めします。
- (Ephemeral Diffie-HellmanPFS) や DHE (Elliptic Curve Ephemeral Diffie-Hellman) などの完全前方秘匿性 ECDHE () を備えた暗号スイート。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。

さらに、 リクエストは、 IAMプリンシパルに関連付けられたアクセスキー ID と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。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて責任共有モデルに従います。 AWS サービスセキュリティ情報については、AWS 「サービスセキュリティドキュメント」ページとAWS 、 AWS コンプライアンスプログラム によるコンプライアンスの取り組みの対象となるサービスを参照してください。

Amazon S3 暗号化クライアントの移行

このトピックでは、アプリケーションで使用している Amazon Simple Storage Service (Amazon S3) 暗号化クライアントをバージョン 1 (V1) からバージョン 2 (V2) に移行し、移行プロセス全体でアプリケーションの可用性を確保する方法について説明します。

移行の概要

この移行は 2 つのフェーズから構成されます。

1. 新しいフォーマットを読み取るために既存のクライアントを更新します。まず、更新されたバージョンの AWS SDK for PHP をアプリケーションにデプロイします。これにより、既存の V1 暗号化クライアントが、新しい V2 クライアントによって書き込まれたオブジェクトを復号できるようになります。アプリケーションで複数の を使用している場合は AWS SDKs、それぞれをSDK個別にアップグレードする必要があります。

2. 暗号化および復号クライアントを V2 に移行します。すべての V1 暗号化クライアントが新しいフォーマットを読み取ることができるようになったら、既存の暗号化および復号化クライアントをそれぞれの V2 バージョンに移行できます。

新しいフォーマットを読み取るために既存のクライアントを更新する

V2 暗号化クライアントは、古いバージョンのクライアントではサポートされていない暗号化アルゴリズムを使用します。移行の最初のステップは、V1 復号クライアントを最新のSDKリリースに更新することです。このステップを完了すると、アプリケーションの V1 クライアントが V2 暗号化クライアントによって暗号化されたオブジェクトを復号できるようになります。 AWS SDK for PHPの各メジャーバージョンについては、以下の詳細を参照してください。

AWS SDK for PHP バージョン 3 のアップグレード

バージョン 3 は AWS SDK for PHPの最新バージョンです。この移行を完了するには、バージョン 3.148.0 以降の `aws/aws-sdk-php` パッケージを使用する必要があります。

コマンドラインからインストールする

Composer を使用してインストールされたプロジェクトの場合、Composer ファイルでパッケージをのバージョン 3.148.0 SDKに更新しSDK、次のコマンドを実行します。

```
composer update aws/aws-sdk-php
```

Phar または Zip ファイルを使用したインストール

次のいずれかの方法を使用します。更新されたSDKファイルは、必ず、require ステートメントによって決定されるコードで必要な場所に配置してください。

Phar ファイルを使用してインストールされたプロジェクトについては、更新されたファイル (`aws.phar`) をダウンロードします。

```
<?php
  require '/path/to/aws.phar';
```

```
?>
```

Zip ファイルを使用してインストールされたプロジェクトについては、更新されたファイル () をダウンロードします。

```
<?php
  require '/path/to/aws-autoloader.php';
?>
```

暗号化および復号クライアントを V2 に移行する

クライアントを更新して新しい暗号化形式を読み取るようにした後で、V2 暗号化および復号化クライアントを使用するようにアプリケーションを更新できます。次の手順は、コードを V1 から V2 に正常に移行する方法を示しています。

V2 クライアントへの更新の要件

1. AWS KMS 暗号化コンテキストは、 S3EncryptionClientV2::putObjectおよび S3EncryptionClientV2::putObjectAsyncメソッドに渡される必要があります。 AWS KMS 暗号化コンテキストは、キーと値のペアの関連付け配列であり、 AWS KMS キー暗号化のために暗号化コンテキストに追加する必要があります。追加のコンテキストが必要ない場合は、空の配列を渡すことができます。

2. @SecurityProfile は S3EncryptionClientV2 の getObject および getObjectAsync メソッドに渡す必要があります。@SecurityProfile は getObject... メソッドの新しい必須パラメータです。'V2' に設定すると、V2 互換形式で暗号化されたオブジェクトのみ復号化できます。また、このパラメータを 'V2_AND_LEGACY' に設定すると、V1 互換形式で暗号化されたオブジェクトを復号化することができます。移行をサポートするために、@SecurityProfile を 'V2_AND_LEGACY' に設定します。'V2' は、新しいアプリケーションの開発でのみ使用します。

3. (オプション) @KmsAllowDecryptWithAnyCmk パラメータを S3EncryptionClientV2::getObject と S3EncryptionClientV2::getObjectAsync* methods. に含め、@KmsAllowDecryptWithAnyCmk という新しいパラメータを追加しました。このパラメータを に設定すると、KMSキーを指定せずに復号化trueが有効になります。デフォルト値は false です。

4. V2 クライアントで復号化する場合、"getObject..." メソッド呼び出しで @KmsAllowDecryptWithAnyCmk パラメータが true に設定されていない場合、KmsMaterialsProviderV2 のコンストラクタに kms-key-id を指定する必要があります。

移行の例

例 1: V2 クライアントへの移行

移行前

```
use Aws\S3\Crypto\S3EncryptionClient;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClient(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

移行後

```
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClientV2(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

例 2: AWS KMS で を使用する kms-key-id

> **Note**
>
> これらの例では、例 1 で定義されたインポートと変数を使用しています。例えば、`$encryptionClient` と指定します。

移行前

```
use Aws\Crypto\KmsMaterialsProvider;
```

```
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProvider(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
    'Cipher' => 'gcm',
    'KeySize' => 256,
];

$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

$result = $encryptionClient->getObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

移行後

```
use Aws\Crypto\KmsMaterialsProviderV2;
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV2(
    new KmsClient([
        'profile' => 'default',
```

```
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
    'Cipher' => 'gcm',
    'KeySize' => 256,
];

$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => true,
    '@SecurityProfile' => 'V2_AND_LEGACY',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

AWS SDK for PHP バージョン 3 に関するよくある質問

クライアントではどのようなメソッドを使用できますか?

AWS SDK for PHP では、サービス記述と動的 __call() マジックメソッドを使用して、API オペレーションが実行されます。ウェブサービスのクライアントで使用可能なメソッドの一覧については、クライアントの API ドキュメントを参照してください。

cURL の SSL 証明書エラーにはどのように対処すればいいですか?

この問題は、古くなった CA バンドルを cURL および SSL で使用した場合に発生する可能性があります。この問題を回避するには、サーバーの CA バンドルを更新するか、または新しい CA バンドルを cURL ウェブサイトから直接ダウンロードします。

デフォルトでは、PHP のコンパイル時に設定されていた CA バンドルが AWS SDK for PHP で使用されます。PHP で使用されるデフォルトの CA バンドルを変更するには、PHP.ini 構成設定で `openssl.cafile` を CA ディスク上の CA ファイルのパスに変更します。

クライアントではどの API バージョンを使用できますか?

クライアントの作成時には `version` オプションが必要です。使用可能な API バージョンの一覧については、各クライアントの API ドキュメントのページ ::aws-php-class:<index.html> を参照してください。特定の API バージョンをロードできない場合は、AWS SDK for PHP のコピーの更新が必要になることがあります。

「version」の設定値に文字列 `latest` を指定すると、クライアントの API プロバイダで見つかった (デフォルトの api_provider では、API モデルの SDK の `src/data` ディレクトリがスキャンされる)、使用可能な最新の API バージョンが使用されます。

> **Warning**
>
> API の更新が含まれている新しいマイナーバージョンの SDK を取り込むと本稼働アプリケーションが中断される可能性があるため、本稼働アプリケーションで `latest` を使用することはお勧めしません。

クライアントではどのリージョンのバージョンを使用できますか?

region オプションは、クライアントの作成時に必要であり、文字列値を使用して指定します。使用可能な AWS リージョンとエンドポイントの一覧については、AWS 全般のリファレンス の「AWS リージョンとエンドポイント」を参照してください。

```
// Set the Region to the EU (Frankfurt) Region.
$s3 = new Aws\S3\S3Client([
    'region'  => 'eu-central-1',
    'version' => '2006-03-01'
]);
```

サイズが 2 GB を越えるファイルをアップロードおよびダウンロードできないのはなぜですか?

PHP の整数型は符号付き整数であり、多くのプラットフォームでは 32 ビット整数が使用されているため、AWS SDK for PHP は 32 ビットスタック (CPU、OS、ウェブサーバー、PHP バイナリを含む) では 2 GB を越えるサイズのファイルを正しく処理できません。これは、よく知られている PHP の問題です。Microsoft Windows の場合は、PHP 7 のビルドでのみ 64 ビット整数がサポートされています。

解決策として、最新バージョンの PHP がインストールされている 64 ビット Linux スタック (64 ビット Amazon Linux AMI など) を使用することをお勧めします。

詳細については、「PHP filesize: 返り値」を参照してください。

送信されるデータはどのようにして確認できますか?

クライアントのコンストラクタで debug オプションを使用すると、送信されるデータなどのデバッグ情報を取得できます。このオプションを true に設定すると、実行されるコマンドの変異、送信されるリクエスト、受信されたレスポンス、および処理された結果がすべて STDOUT に出力されます。その出力には、ネットワーク経由で送受信されるデータも含まれています。

```
$s3Client = new Aws\S3\S3Client([
    'region'  => 'us-standard',
    'version' => '2006-03-01',
    'debug'   => true
]);
```

リクエストに任意のヘッダーを設定するにはどうすればいいですか?

Aws\HandlerList または Aws\CommandInterface の Aws\ClientInterface にカスタムミドルウェアを追加することによって、サービスオペレーションに任意のヘッダーを追加できます。次の例では、Aws\Middleware::mapRequest ヘルパーメソッドを使用して、Amazon S3 の特定の PutObject オペレーションに X-Foo-Baz ヘッダーを追加する方法を示しています。

詳細については、「mapRequest」を参照してください。

任意のリクエストに署名するにはどうすればいいですか?

SDK の :aws-php-class: SignatureV4 class <class-Aws.Signature.SignatureV4.html> を使用して、任意の :aws-php-class: PSR-7 request <class-Psr.Http.Message.RequestInterface.html> に署名できます。

署名の例については、「AWS SDK for PHP バージョン 3 によるカスタム Amazon CloudSearch ドメインリクエストへの署名」を参照してください。

送信する前にコマンドを変更するにはどうすればいいですか?

Aws\HandlerList または Aws\CommandInterface の Aws\ClientInterface にカスタムミドルウェアを追加することによって、コマンドを送信する前にそのコマンドを変更できます。次の例では、送信する前にコマンドにカスタムコマンドパラメータを追加する (基本的にはデフォルトのオプションを追加する) 方法を示しています。この例では、Aws\Middleware::mapCommand ヘルパーメソッドを使用しています。

詳細については、「mapCommand」を参照してください。

CredentialsException とは何ですか?

AWS SDK for PHP の使用時に Aws\Exception\CredentialsException が発生する場合は、どの認証情報でも SDK が指定されていないために、SDK が環境内で認証情報を見つけることができなかったことを意味しています。

認証情報を使用せずにクライアントをインスタンス化する場合、サービスオペレーションの初回実行時に SDK は認証情報を見つけようとします。SDK は、特定の環境変数をチェックした後に、

設定済みの Amazon EC2 インスタンスでのみ利用できるインスタンスプロファイルの認証情報を調べます。認証情報がまったく指定されていないかまたは見つからない場合に、`Aws\Exception\CredentialsException` がスローされます。

このエラーが発生した場合にインスタンスプロファイルの認証情報を使用するには、SDK が実行されている Amazon EC2 インスタンスに適切な IAM ロールを設定しておく必要があります。

このエラーが発生した場合にインスタンスプロファイルの認証情報を使用しない場合は、SDK に認証情報を適切に指定しておく必要があります。

詳細については、「AWS SDK for PHP バージョン 3 の認証情報」を参照してください。

AWS SDK for PHP は HHVM で動作しますか?

現時点では、AWS SDK for PHP は HHVM では実行されず、HHVM での yield セマンティクスに関する問題が解決されるまで実行できません。

SSL を無効にするにはどうすればいいですか?

クライアントのファクトリメソッドの `scheme` パラメータを「http」に設定することによって、SSL を無効にできます。一部のサービスでは `http` アクセスがサポートされていないことに注意する必要があります。リージョン、エンドポイント、およびサポートされているスキームの一覧については、AWS 全般のリファレンス の「AWS のリージョンとエンドポイント」を参照してください。

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region'  => 'us-west-2',
    'scheme'  => 'http'
]);
```

> **Warning**
>
> SSL ではすべてのデータが暗号化される必要があり、接続ハンドシェイクを完了するために TCP のみの場合より多くの TCP パケットが必要であるため、SSL を無効にするとパフォーマンスが多少改善される可能性があります。ただし、SSL が無効になっている場合、すべてのデータは暗号化されずにネットワーク経由で送信されます。SSL を無効にする前に、ネットワークでのセキュリティへの影響と傍受の可能性を注意深く検討する必要があります。

「解析エラー」についてはどう対処すればいいですか?

PHP エンジンは、理解できない構文を検出したときに解析エラーをスローします。異なるバージョンの PHP 用に記述されたコードを実行しようとすると、ほとんどの場合にこのエラーが発生します。

解析エラーが発生した場合は、システムが AWS SDK for PHP バージョン 3 の要件と推奨事項を満たしていることを確認します。

Amazon S3 クライアントが gzip で圧縮されたファイルを解凍するのはなぜですか?

デフォルトの Guzzle 6 HTTP ハンドラーなどの一部の HTTP ハンドラーは、デフォルトでは、圧縮されたレスポンス本文を解凍します。decode_content HTTP オプションを `false` に設定することによって、この動作をオーバーライドできます。下位互換性のために、このデフォルトは変更できませんが、S3 クライアントレベルでのコンテンツのデコードを無効にすることをお勧めします。

コンテンツの自動デコードを無効にする方法の例については、「decode_content」を参照してください。

Amazon S3 での本文署名を無効にするにはどうすればよいですか?

コマンドオブジェクトの `ContentSHA256` パラメーターを `Aws\Signature\S3SignatureV4::UNSIGNED_PAYLOAD` に設定することによって、本文署名を無効にすることができます。そうすると、AWS SDK for PHP では、その値が正規リクエストの「x-amz-content-sha-256」ヘッダーおよび本文チェックサムとして使用されます。

```
$s3Client = new Aws\S3\S3Client([
    'version' => '2006-03-01',
    'region'  => 'us-standard'
]);

$params = [
    'Bucket' => 'foo',
    'Key'    => 'baz',
    'ContentSHA256' => Aws\Signature\S3SignatureV4::UNSIGNED_PAYLOAD
];

// Using operation methods creates command implicitly
```

```
$result = $s3Client->putObject($params);

// Using commands explicitly.
$command = $s3Client->getCommand('PutObject', $params);
$result = $s3Client->execute($command);
```

AWS SDK for PHP では再試行スキームはどのように処理されますか?

AWS SDK for PHP には、再試行動作を処理する `RetryMiddleware` があります。サーバーエラーの HTTP ステータスコード 5xx で言えば、この SDK は 500、502、503、504 のステータスコードの場合に再試行します。

`RequestLimitExceeded`、`Throttling`、`ProvisionedThroughputExceededException`、`Throttl` などのスロットリング例外も再試行で処理されます。

AWS SDK for PHP では、再試行間隔を指数関数的に増やす手法やジッター低減アルゴリズムも再試行スキームに組み込まれています。さらに、デフォルトの再試行動作は、Amazon DynamoDB (`10` に設定) 以外のすべてのサービスで `3` に設定されています。

エラーコードがある例外を処理するにはどうすればいいですか?

AWS SDK for PHP でカスタマイズされている `Exception` クラスに加えて、AWS の各サービスには AwsExceptionAwsException から継承した独自の例外クラスがあります。より具体的なエラータイプを特定すると、各メソッドの `Errors` セクションに一覧が記載されている API 固有のエラーをキャッチできます。

エラーコードの情報は、 の getAwsErrorCode()`Aws\Exception\AwsException` を使用して取得できます。

```
$sns = new \Aws\Sns\SnsClient([
    'region' => 'us-west-2',
    'version' => 'latest',
]);

try {
    $sns->publish([
        // parameters
        ...
```

```
    ]);
    // Do something
} catch (SnsException $e) {
    switch ($e->getAwsErrorCode()) {
        case 'EndpointDisabled':
        case 'NotFound':
            // Do something
            break;
    }
}
```

用語集

API バージョン

サービスには、1 つ以上の API バージョンがあり、使用しているバージョンの種類で、有効なオペレーションとパラメーターの種類が決まります。API バージョンは、日付のようなフォーマットです。例えば、Amazon S3 の最新 API バージョンは、`2006-03-01` となります。クライアントオブジェクトを設定するときに、バージョンを指定します。

クライアント

クライアントオブジェクトは、サービスのオペレーションを実行するために使用されます。SDK でサポートされている各サービスには、対応するクライアントオブジェクトがあります。クライアントオブジェクトには、サービスのオペレーションと 1 対 1 で対応するメソッドがあります。クライアントオブジェクトの作成および使用方法の詳細については「基本使用法ガイド」を参照してください。

コマンド

コマンドオブジェクトはオペレーションの実行をカプセル化します。SDK の基本使用パターンに従うと、コマンドオブジェクトを直接扱いません。同時リクエストとバッチ処理のような SDK の高度な機能を使用するには、クライアントの `getCommand()` メソッドを使用してコマンドオブジェクトにアクセスできます。詳細については、「AWS SDK for PHP バージョン 3 でのコマンドオブジェクト」ガイドを参照してください。

Handler

ハンドラーは、コマンドおよびリクエストから結果への実際の変換を実行する関数です。ハンドラーは、通常、HTTP リクエストを送信します。ハンドラーをミドルウェアで構成して動作を補強できます。ハンドラー関数は、`Aws\CommandInterface` と `Psr\Http\Message\RequestInterface` を受け入れて、`Aws\ResultInterface` で満たすかまたは `Aws\Exception\AwsException` 理由で拒否する promise を返します。

JMESPath

JMESPath は JSON 類似データ向けのクエリ言語です。AWS SDK for PHP は JMESPath 式を使用して PHP のデータ構造をクエリします。JMESPath 式は `Aws\Result` と `Aws\ResultPaginator` オブジェクトで `search($expression)` メソッドを介して直接使用できます。

ミドルウェア

ミドルウェアは、コマンドの転送動作を補強して「next」ハンドラーに委任する、特殊なタイプの高レベル関数です。ミドルウェア関数は `Aws\CommandInterface` と `Psr\Http\Message\RequestInterface` を受け入れて、`Aws\ResultInterface` で満たすかまたは `Aws\Exception\AwsException` 理由で拒否する promise を返します。

操作

サービスの API にある単一のオペレーションを指します (DynamoDB では `CreateTable`、Amazon EC2 では `RunInstances` など)。SDK では、オペレーションは対応するサービスのクライアントオブジェクトで同じ名前のメソッドを呼び出すことによって実行されます。オペレーションの実行には、サービスへの HTTP リクエストの作成と送信、応答の解析があります。オペレーションのこの実行プロセスは、コマンドオブジェクト経由で SDK により抽象化されます。

ページネーター

一部の AWS サービスオペレーションではページ分割され、結果が一部切り捨てられて応答します。例えば、Amazon S3 の `ListObjects` オペレーションが一度に返せるのは、1000 個のオブジェクトまでです。このようなオペレーションでは後続のリクエストが必要になります。トークン (マーカー) パラメーターを指定して結果セット全体を取得します。ページネーターは SDK の機能で、開発者がページ分割 API を簡単に使用できるようにこのプロセスを抽象化します。クライアントの `getPaginator()` メソッドを使用してアクセスします。詳細については、「AWS SDK for PHP バージョン 3 でのページネーター」ガイドを参照してください。

promise

promise は、非同期オペレーションの最終結果を表します。promise とのやり取りの主な手段は、メソッドを使用するもので、promise の最終結果の値、または promise が実行できない理由を受け取るコールバックを登録します。

リージョン

サービスは、1 か所以上の地理的リージョンでサポートされています。サービスは各リージョンで異なるエンドポイント/URL を持つことができます。これはアプリケーションのデータレイテンシーを軽減するために存在します。クライアントオブジェクトを設定するときは、リージョンを指定します。これにより、SDK でサービスに使用するエンドポイントを特定できます。

SDK

「SDK」という用語は AWS SDK for PHP ライブラリ全体を指すこともありますが、`Aws\Sdk` クラス (docs) の場合もあります。各サービスのクライアントオブジェクトに対してファクトリとし

て機能します。Sdk クラスにより、それが作成するすべてのクライアントオブジェクトに適用されるグローバル設定値のセットが提供できます。

サービス

任意の AWS サービス (Amazon S3、Amazon DynamoDB、AWS OpsWorks など) を参照する一般的な方法です。各サービスには、対応するクライアントオブジェクトが SDK にあり、1 つ以上の API バージョンをサポートします。また各サービスには、その API を構成する 1 つ以上のオペレーションがあります。サービスは、1 か所以上のリージョンでサポートされています。

署名

オペレーションを実行すると、SDK は認証情報を使用して、リクエストのデジタル署名を作成します。このサービスは、リクエストを処理する前に、署名を確認します。署名プロセスは、SDK でカプセル化され、クライアントに設定した認証情報を使用して自動的に実行されます。

ウェーター

ウェーターは SDK の機能で、これによりリソースの状態を変更する、事実上結果整合性があるまたは非同期であるオペレーションとの連携が簡単になります。例えば、Amazon DynamoDB `CreateTable` オペレーションはすぐに応答を返しますが、数秒経たないとテーブルにアクセスする準備ができていないことがあります。ウェーターを実行すると、リソースのステータスのポーリングおよびスリープにより、リソースが特定の状態になるまで待機できます。ウェーターは、クライアントの `waitUntil()` メソッドを使用してアクセスします。詳細については、「AWS SDK for PHP バージョン 3 でのウェーター」ガイドを参照してください。

最新の AWS の用語については、「AWS 全般のリファレンス」の「AWS 用語集」を参照してください。

ドキュメント履歴

次の表は、AWS SDK for PHP デベロッパーガイドの前回リリースからの重要な変更点をまとめたものです。

最新の変更:

変更	説明	日付
Amazon EventBridge グローバルエンドポイント	Amazon EventBridge グローバルエンドポイントの使用方法を示すコード例を追加する	2023 年 12 月 22 日
AWS 共通ランタイム (AWS CRT)	SDK for PHP による AWS Common Runtime (AWS CRT) の使用について説明するトピックを追加します。	2023 年 11 月 17 日
StreamWrapper mkdir() の更新	`mkdir()` を使用した、バケットとフォルダオブジェクトの操作方法に関する情報を追加します。	2023 年 11 月 2 日
サービスクライアントの作成	「latest」がデフォルトなので、「version」パラメータを削除してコードスニペットを更新します。	2023 年 8 月 31 日
目次	目次を更新して、コード例をよりわかりやすくしました。	2023 年 6 月 1 日
IAM ベストプラクティスの更新	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。開始方法の更新	2023 年 5 月 20 日

Amazon S3 Transfer Manager	`add_content_md5` 転送オプションが追加されました。	2023 年 4 月 13 日
Amazon S3 マルチパートアップロード	同期アップロードの設定情報を含めました。非同期アップロード用の `add_content_md5` アップロードオプションが追加されました。	2023 年 4 月 13 日
リファレンス情報	AWS SDK およびツールリファレンスガイドの関連する詳細コンテンツへのリンクを複数追加しました。ガイドのフォーマットを更新しました。	2022 年 9 月 14 日
一般的なクリーンアップ	AWS SDK およびツールリファレンスガイドへの参照を追加しました。用語の更新を反映するように AWS Key Management Service セクションを更新しました。	2022 年 8 月 23 日
AWS サービスの使用	で使用できるコード例のリストが含まれています GitHub。	2022 年 4 月 1 日
SDK メトリクスの有効化	2021 年 12 月 20 日に廃止された SDK メトリクスの有効化に関する情報を削除しました。	2022 年 1 月 27 日
Amazon S3 暗号化クライアントの移行	Amazon S3 暗号化クライアントの移行に関するトピックを追加しました	2020 年 8 月 7 日

古い変更:

変更	説明	リリース日
Secrets Manager の例	サービスの例の追加	2019 年 3 月 27 日
エンドポイント検出	エンドポイント検出の設定	2019 年 2 月 15 日
Amazon CloudFront	サービスの例の追加	2019 年 1 月 25 日
サービスの機能	SDK のメトリクス	2018 年 1 月 11 日
Amazon Kinesis、Amazon SNS	サービスの例の追加	2018 年 12 月 14 日
Amazon SES の例	サービスの例の追加	2018 年 10 月 5 日
AWS KMS の例	サービスの例の追加	2018 年 8 月 8 日
認証情報	認証情報ガイドの明確化と簡素化	2018 年 6 月 30 日
MediaConvert 例	サービスの例の追加	2018 年 6 月 15 日
新しいウェブレイアウト	ドキュメントの AWS スタイルへの切り替え	2018 年 5 月 9 日
Amazon S3 の暗号化	クライアント側の暗号化	2017 年 11 月 17 日
Amazon S3、Amazon SQS	サービスの例の追加	2017 年 3 月 26 日
Amazon S3、IAM、Amazon EC2	サービスの例の追加	2017 年 3 月 17 日
認証情報の追加	AssumeRole および ini のサポートを追加	2017 年 1 月 17 日
S3 の例	S3 マルチリージョンと署名付き投稿	2016 年 3 月 18 日
OpenSearch サービスと Amazon CloudSearch	サービスの例の追加	2015 年 12 月 28 日

変更	説明	リリース日
コマンドライン	コマンドパラメータの追加	2015 年 8 月 13 日
サービスの機能	S3 と AWS に関するサービスの機能の追加	2015 年 4 月 30 日
SDK の新バージョン	AWS SDK for PHP のバージョン 3 をリリースしました。	2015 年 5 月 26 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。