バージョン 3 のデベロッパーガイド

AWS SDK for PHP

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS SDK for PHP: バージョン 3 のデベロッパーガイド

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスはAmazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

AWS SDK for PHPとは 1
SDK の使用を開始する 1
追加リソース 1
API ドキュメント 2
SDK メジャーバージョンのメンテナンスとサポート 2
入門 3
前提条件 3
要件 3
レコメンデーション 3
互換性テスト 5
インストール 5
Composer 経由で依存関係 AWS SDK for PHP として をインストールする 5
パッケージ済み phar を使用したインストール 6
ZIP ファイルを使用したインストール 7
による認証 AWS 7
ローカル開発用に IAM アイデンティティセンターで認証を設定する 8
AWS アクセスポータルセッションを開始する 9
認証の詳細 9
シンプルなアプリケーションの作成 10
前提条件 10
コードで SDK をインクルードする 10
コードを書き込む 11
プログラムの実行 11
次のステップ 12
SDK AWS Cloud9 で を使用する 12
ステップ 1: を使用する AWS アカウント ように を設定する AWS Cloud9 13
ステップ 2: AWS Cloud9 開発環境を設定する 13
ステップ 3: をセットアップする AWS SDK for PHP 14
ステップ 4: サンプルコードをダウンロードする 15
ステップ 5: コードサンプルの実行 15
サービスクライアントの設定 17
外部でのクライアント設定 17
クライアント設定の設定プロバイダーチェーン 18
外部設定を使用して設定されたサービスクライアントを作成する 18

AWS SDK for PHP バージョン 3 環境変数 ... 19
コード内のクライアント設定 ... 20
コードの基本設定 ... 20
Sdk クラスの使用 ... 20
コンストラクタオプション ... 22
AWS リージョン ... 50
リージョン解決チェーン ... 51
ベストプラクティス ... 52
認証情報プロバイダ ... 52
AWS SDK for PHP バージョン 3 の認証情報プロバイダーとは ... 53
AWS SDK for PHP バージョン 3 のデフォルトの認証情報プロバイダーチェーンについて ... 54
AWS SDK for PHP バージョン 3 の組み込み認証情報プロバイダー ... 55
SDK for PHP での認証情報プロバイダーの連鎖 ... 65
SDK for PHP で使用するカスタム認証情報プロバイダーの作成 ... 65
SDK for PHP での認証情報のメモリ ... 66
IAM ロールの継承 ... 67
SDK for PHP AWS STS で から一時的な認証情報を使用する ... 74
SDK for PHP で匿名クライアントを作成する ... 77
CRT AWS 拡張機能を使用する ... 77
CRT AWS 拡張機能が必要ですか？ ... 77
CRT AWS 拡張機能をインストールするにはどうすればよいですか？ ... 78
SDK の使用 ... 79
リクエストの実行 AWS のサービス ... 79
SDK リクエストワークフローの概要 ... 79
基本的なサービスクライアントの作成 ... 80
リクエストの実行 ... 81
結果オブジェクトの使用 ... 82
コマンドオブジェクト ... 84
非同期プログラミング ... 91
promise ... 93
エラー処理 ... 100
同期エラー処理 ... 100
非同期エラー処理 ... 101
ハンドラーとミドルウェア ... 102
ハンドラー ... 102
ミドルウェア ... 104

カスタムハンドラーの作成 ... 112
Streams ... 113
ストリームデコレータ ... 113
Pagination (ページ分割) ... 118
ページネーターオブジェクト ... 118
結果からのデータの列挙 ... 119
非同期ページ割り ... 120
ウェイター ... 121
ウェーターの設定 ... 121
非同期の待機 ... 123
JMESPath 式 ... 124
結果からのデータの抽出 ... 125
ページネーターからのデータの抽出 ... 129
呼び出し AWS のサービス ... 131
機能とオプションを使用する ... 131
Amazon DynamoDB ... 131
Amazon S3 ... 138
ガイダンス付きのコード例 ... 161
認証情報 ... 161
Amazon CloudFront の例 ... 162
Amazon CloudSearch ... 191
Amazon CloudWatch の例 ... 194
Amazon EC2 の例 ... 218
Amazon OpenSearch Service ... 232
AWS Identity and Access Management 例 ... 233
AWS Key Management Service ... 258
Kinesis での例 ... 280
AWS Elemental MediaConvert ... 297
Amazon S3の例 ... 304
AWS Secrets Manager ... 338
Amazon SES の例 ... 347
Amazon SNS の例 ... 380
Amazon SQS の例 ... 399
Amazon EventBridge ... 412
コードサンプル ... 414
API Gateway ... 415

アクション 415
シナリオ 421
Aurora 421
シナリオ 421
Auto Scaling 422
基本 424
アクション 415
Amazon Bedrock 438
アクション 415
Amazon Bedrock ランタイム 440
シナリオ 421
Amazon Nova 442
Amazon Titan Image Generator 444
Anthropic Claude 445
Stable Diffusion 447
Amazon DocumentDB 448
サーバーレスサンプル 449
DynamoDB 450
基本 424
アクション 415
シナリオ 421
サーバーレスサンプル 449
Amazon EC2 483
アクション 415
AWS Glue 489
基本 424
アクション 415
IAM 509
基本 424
アクション 415
Kinesis 527
サーバーレスサンプル 449
AWS KMS 530
基本 424
アクション 415
Lambda 567

基本 ... 424
アクション ... 415
シナリオ ... 421
サーバーレスサンプル ... 449
Amazon MSK ... 597
サーバーレスサンプル ... 449
Amazon RDS ... 599
アクション ... 415
シナリオ ... 421
サーバーレスサンプル ... 449
Amazon RDS データサービス ... 607
シナリオ ... 421
Amazon Rekognition ... 609
シナリオ ... 421
Amazon S3 ... 610
基本 ... 424
アクション ... 415
シナリオ ... 421
サーバーレスサンプル ... 449
S3 ディレクトリバケット ... 633
基本 ... 424
Amazon SES ... 649
シナリオ ... 421
Amazon SNS ... 650
アクション ... 415
シナリオ ... 421
サーバーレスサンプル ... 449
Amazon SQS ... 671
サーバーレスサンプル ... 449
バージョン 2 からの移行 ... 675
序章 ... 675
バージョン 3 の新機能 ... 675
分離された HTTP レイヤー ... 675
非同期リクエスト ... 91
バージョン 2 との違いは何ですか？ ... 676
プロジェクトの依存関係が更新される ... 676

リージョンとバージョンオプションが必要になりました ... 677
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する ... 677
クライアント設定が変更されました ... 678
一部の API の結果が変更されました ... 679
列挙型クラスが削除されました ... 682
きめ細かな例外クラスが削除されました ... 682
静的ファサードクラスが削除されました ... 683
ページネーターはイテレーターよりも優先されます ... 683
多くの上位レベルの抽象化が変更されました ... 684
SDK の両バージョンからのコードサンプルの比較 ... 685
例: Amazon S3 `ListObjects`オペレーション ... 685
例: グローバル設定でクライアントをインスタンス化する ... 687
セキュリティ ... 689
データ保護 ... 689
Identity and Access Management ... 690
対象者 ... 691
アイデンティティを使用した認証 ... 691
ポリシーを使用したアクセスの管理 ... 695
IAM AWS のサービス の操作方法 ... 698
AWS ID とアクセスのトラブルシューティング ... 698
コンプライアンス検証 ... 700
耐障害性 ... 701
インフラストラクチャセキュリティ ... 702
Amazon S3 暗号化クライアントの移行 ... 702
移行の概要 ... 703
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 ... 703
暗号化および復号クライアントを V2 に移行する ... 704
移行の例 ... 705
よくある質問 ... 708
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? ... 708
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? ... 708
クライアントではどの API バージョンを使用できますか? ... 708
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? ... 709
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? ... 709
送信されるデータはどのようにして確認できますか? ... 709

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? 710
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? 710
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? 710
CredentialsException とは何ですか? 710
は HHVM で AWS SDK for PHP 動作しますか ? 711
SSL を無効にするにはどうすればいいですか? 711
「解析エラー」についてはどう対処すればいいですか? 712
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? 712
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? 712
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? 713
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? 713
用語集 715
ドキュメント履歴 718
..... dccxxiii

AWS SDK for PHP バージョン 3 とは

AWS SDK for PHP バージョン 3 を使用すると、PHP 開発者は 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と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。で現在サポートされているサービスを確認するには AWS SDK for PHP、「サービス名と API バージョン」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、AWS SDK for PHP バージョン 3 の開始方法 の章に従ってください。を使用した認証 AWS、開発環境の設定、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説明します。

追加リソース

- よくある質問
- 用語集
- AWS SDKsおよびツールリファレンスガイド: AWS SDKs。
- Guzzle のドキュメント
- を使用したコード例は AWS SDK for PHP 、awsdocs/aws-doc-sdk-examples リポジトリで入手できます。
- Gitter の PHP SDK コミュニティ。
- AWS re:Post.

GitHub:

- のソースコード AWS SDK for PHP は、aws/aws-sdk-php リポジトリで入手できます。
- 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 ドキュメントについては、https://docs.aws.amazon.com/sdk-for-php/latest/reference/ を参照してください。

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

AWS SDK for PHP バージョン 3 の開始方法

この章では、 AWS SDK for PHP バージョン 3 の起動と実行について説明します。

トピック

- 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 バージョン 3 AWS を使用した での認証
- AWS SDK for PHP バージョン 3 を使用したシンプルなアプリケーションの作成
- AWS SDK for PHP バージョン 3 AWS Cloud9 で を使用する

AWS SDK for PHP バージョン 3 の要件と推奨事項

で最良の結果を得るには AWS SDK for PHP、環境が以下の要件と推奨事項をサポートしていることを確認してください。

要件

を使用するには AWS SDK for PHP、SimpleXML PHP 拡張機能が有効になっている PHP バージョン 8.1 以降を使用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 を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 8.1 以降を使用していることを確認してください。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 経由で依存関係 AWS SDK for PHP として 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 経由で依存関係 AWS SDK for PHP として を追加する

Composer がシステムに既にグローバルにインストールされている場合は、プロジェクトのベースディレクトリで以下を実行して、依存関係 AWS SDK for PHP として 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 AWS SDK for PHP で を利用するには、次のようにスクリプトにオートローダーを含めます。

```
<?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

の各リリース AWS SDK for PHP には、SDK の実行に必要なすべてのクラスと依存関係を含むパッケージ化された phar (PHP アーカイブ) が含まれています。さらに、phar は AWS SDK for PHP とそのすべての依存関係のクラスオートローダーを自動的に登録します。

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

には、SDK の実行に必要なすべてのクラスと依存関係を含む ZIP ファイル AWS SDK for PHP が含まれています。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 バージョン 3 AWS を使用した での認証

で開発 AWS するときに、コードが で認証される方法を確立する必要があります AWS のサービス。AWS リソースへのプログラムによるアクセスは、環境と利用可能な 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。

ローカル開発用に IAM アイデンティティセンターで認証を設定する

ローカルで開発中で、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、 を設定することをお勧めします AWS IAM Identity Center。この方法には、設定を容易に AWS CLI し、 AWS アクセスポータルに定期的にサインインするための のインストールが含まれます。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 で参照できる設定値のセットを持つ`[default]`プロファイルを持つ共有 AWSconfig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。`sso_role_name` 値は、IAM Identity Center アクセス許可セットに接続された IAM ロールであり、アプリケーションで AWS のサービス 使用されている へのアクセスを許可します。

  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 認証を使用するには、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

アクセスするアプリケーションを実行する前に AWS のサービス、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。

アクティブなセッションが既に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`、認証情報を指定する必要はありません。
> サインインプロセスにより、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可能性があります。

認証の詳細

- 認証に IAM アイデンティティセンターを使用する方法の詳細については、 SDK およびツールリファレンスガイドの「IAM アイデンティティセンター認証を理解する」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 を使用したシンプルなアプリケーションの作成

を使用して Amazon S3 に挨拶します AWS SDK for PHP。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。

前提条件

- SDK をダウンロードしてインストールする
- を使用する前に AWS SDK for PHP、 で認証を設定する必要があります AWS。認証の設定の詳細については、「AWS SDK for PHP バージョン 3 AWS を使用した での認証」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。

コードを書き込む

認証できることを確認します。

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。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

/**
 * List your Amazon S3 buckets.
 */

//Create a S3Client
// snippet-start:[s3.php.list_buckets.main]
$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

その他の多くの Amazon S3 オペレーションをテストするには、GitHub で AWS のコードサンプルリポジトリを確認してください。

AWS SDK for PHP バージョン 3 AWS Cloud9 で を使用する

> **Note**
>
> AWS Cloud9 は新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS Cloud9 引き続き使用できます。詳細はこちら。

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、構築、実行、テスト、デバッグ、リリースに使用するツールのコレクションが含まれています。AWS Cloud9 で を使用すると AWS SDK for PHP 、ブラウザを使用して PHP コードを記述して実行できます。 には、コードエディタやターミナルなどのツール AWS Cloud9 が含まれています。AWS Cloud9 IDE はクラウドベースのため、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、またはどこからでもプロジェクトに取り組むことができます。の一般的な情報については AWS Cloud9、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 Cloud9 で をセットアップします AWS SDK for PHP。

- ステップ 1: を使用する AWS アカウント ように を設定する AWS Cloud9
- ステップ 2: AWS Cloud9 開発環境を設定する
- ステップ 3: をセットアップする AWS SDK for PHP
- 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: コード例を実行する

ステップ 1: を使用する AWS アカウント ように を設定する AWS Cloud9

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

> **Note**
>
> 認証 AWS IAM Identity Center に を使用している場合は、IAM コンソールのユーザーアタッチポリシー`iam:ListInstanceProfilesForRole`に の必要なアクセス許可を追加する必要がある場合があります。

コンソールにアクセスして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 、AWS Cloud9 「 ユーザーガイド」の AWS Cloud9 「 のチームセットアップ AWS Cloud9」を参照してください。

ステップ 2: AWS Cloud9 開発環境を設定する

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、 はその環境の IDE AWS Cloud9 を開きます。

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、 に環境を作成し、Amazon EC2 インスタンスを起動して、新しいインスタンスを新しい環境に接続する AWS Cloud9 ように指示します。これは、 の使用を開始する最も速い方法です AWS Cloud9。

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で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: をセットアップする AWS SDK for PHP

が開発環境の IDE を AWS Cloud9 開いたら、ターミナルウィンドウを使用して環境で をセットアップ AWS SDK for PHP 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 経由で依存関係 AWS SDK for PHP として を追加する

Composer がシステムに既にグローバルにインストールされている場合は、プロジェクトのベースディレクトリで以下を実行して、依存関係 AWS SDK for PHP として 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 AWS SDK for PHP で を利用するには、次のようにスクリプトにオートローダーを含めます。

```
<?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する

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

公式 AWS SDK ドキュメントで使用されているすべてのコード例のコピーを環境のルートディレクトリに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
```

のコード例は `ENVIRONMENT_NAME/aws-doc-sdk-examples/php` ディレクトリにあります AWS SDK for PHP 。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
```

でファイルを開くには AWS Cloud9、ターミナルウィンドウで `ListBuckets.php`を直接クリックします。

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にある実行ボタンを選択します。 は `.php`ファイル拡張子 AWS Cloud9 を自動的に検出し、PHP (組み込みウェブサーバー) 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: Command. は、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 AWS Cloud9 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: Runner. はファイル拡張子が であることを AWS Cloud9 検出.phpし、コードを実行する PHP (組み込みウェブサーバー) Runner を選択します。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 でのサービスクライアントの設定

プログラムで にアクセスするために AWS のサービス、 AWS SDK for PHP バージョン 3 では各 にクライアントオブジェクトを使用します AWS のサービス。たとえば、アプリケーションが Amazon EC2 にアクセスする必要がある場合は、Ec2Client クラスのインスタンスである Amazon EC2 クライアントオブジェクトを作成して、そのサービスとインターフェイス接続します。 Ec2Client 次に、サービスクライアントを使用して、その AWS のサービスに対してリクエストを実行します。

SDK の動作を設定するには多くの方法がありますが、最終的にはサービスクライアントの動作と関係があります。コードが設定を使用するサービスクライアントを作成するまで、設定は効果がありません。

指定する設定の例は次のとおりです。

- サービスを呼び出す AWS ときにコードが で認証される方法
- サービスクライアント AWS リージョン で使用する 。
- サービス呼び出しの再試行とタイムアウトの設定
- HTTP プロキシ設定

多くの AWS SDKs共通する設定、機能、その他の基本的な概念については、 SDK およびツールリファレンスガイドを参照してください。 AWS SDKs

AWS SDK for PHP バージョン 3 のサービスクライアントを外部で設定する

多くの設定はコードの外部で処理できます。ほとんどの設定は、環境変数として設定することも、別の共有 AWS `config`ファイルで設定することもできます。 AWS 共有`config`ファイルは、プロファイルと呼ばれる個別の設定セットを維持し、環境やテストごとに異なる設定を提供できます。共有ファイル`config`と `credentials` ファイルの詳細については AWS 、 AWS SDKsおよびツールリファレンスガイドを参照してください。

ほとんどの環境変数と共有`config`ファイル設定は標準化され、 AWS SDKsやツール間で共有され、さまざまなプログラミング言語やアプリケーションで一貫した機能をサポートします。

SDK が環境変数または設定ファイルから解決できるすべての設定を確認するには、「 SDK およびツールリファレンスガイド」の「設定リファレンス」を参照してください。 AWS SDKs

クライアント設定の設定プロバイダーチェーン

SDK は、複数の場所 (またはソース) をチェックして設定値を見つけます。

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
   - 環境変数の設定の詳細については、 SDK およびツールリファレンスガイドの「環境変数」を参照してください。 AWS SDKs
   - シェルの環境変数は、システム全体、ユーザー全体、特定のターミナルセッションのさまざまなレベルで設定できることに注意してください。
3. 共有 `config` および `credentials` ファイル
   - これらのファイルの設定の詳細については、 SDK `config`およびツールリファレンスガイドの「共有ファイル」および`credentials`「ファイル」を参照してください。 AWS SDKs
4. SDK ソースコード自体によって提供されるデフォルト値が最後に使用されます。
   - リージョンなどの一部のプロパティにはデフォルトがありません。コード、環境設定、または共有`config`ファイルで明示的に指定する必要があります。SDK が必要な設定を解決できない場合、API リクエストは実行時に失敗する可能性があります。

この一般的な設定チェーンに加えて、 AWS SDK for PHP バージョン 3 では、認証情報プロバイダーチェーンや解決チェーンなどの特殊なプロバイダーチェーンも使用します。 AWS リージョンこれらの特殊なチェーンは、SDK が実行されている環境を考慮したプロバイダーを追加します。たとえば、コンテナまたは EC2 インスタンスの場合です。

外部設定を使用して設定されたサービスクライアントを作成する

と通信するには、アプリケーションにサービスクライアントを作成する必要があります AWS のサービス。サービスクライアントは への重要な接続であり AWS のサービス、複雑な通信の詳細をすべて処理するため、心配する必要はありません。セキュリティ、エラー処理、再試行などの重要なタスクを自動的に処理するため、技術的な複雑さに対処するのではなく、アプリケーションの構築に集中できます。

パラメータなしでコンストラクタを使用してサービスクライアントを構築する

必要なすべての設定が外部ソースからのものである場合は、空のコンストラクタを使用してサービスクライアントを作成できます。

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

use Aws\S3\S3Client;

$s3 = new S3Client([]);
```

前のコードスニペットは`S3Client`インスタンスを作成します。作成時に、SDK は設定プロバイダーチェーンで設定を調べます。SDK が設定値を検出すると、その値が使用されます。

デフォルトの AWS リージョン プロバイダーチェーンとデフォルトの認証情報プロバイダーチェーンは、作成プロセスでも使用されます。チェーンのどこかで、SDK は を解決 AWS リージョン して使用し、リクエストに署名するための認証情報を取得できるようにする設定を見つける必要があります。SDK がこれらの値を見つけられない場合、クライアントの作成は失敗します。

AWS SDK for PHP バージョン 3 環境変数

AWS SDK for PHP バージョン 3 は、ほとんどの SDK でサポートされているクロス SDK 設定を超えて、次の環境変数で動作します。 AWS SDKs

AWS_SDK_LOAD_NONDEFAULT_CONFIG

設定すると、この環境変数は AWS `config`、ファイル () に加えて`credentials`、ファイル (`~/.aws/config`) から認証情報をロードするように SDK に指示します`~/.aws/credential`。

AWS_SDK_UA_APP_ID

SDK によって行われたリクエストの User-Agent ヘッダーに含まれるカスタムアプリケーション識別子を設定します。

AWS_SUPPRESS_PHP_DEPRECATION_WARNING

true に設定すると、SDK が生成する可能性のある PHP 非推奨警告が抑制されます。

AWS SDK for PHP バージョン 3 のコードでサービスクライアントを設定する

外部でサービスクライアントを設定する代わりに、または追加で、プログラムでコードでサービスクライアントを設定できます。

サービスクライアントをコードで設定することで、利用可能な多くのオプションをきめ細かく制御できます。外部で設定できる設定のほとんどは、コードで設定することもできます。

コードの基本設定

オプションの連想配列をクライアントのコンストラクタに渡すことで、コードでサービスクライアントを作成して設定できます。次の例では、連想配列には、クライアントが使用する「リージョン」オプションのみが含まれます。

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

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

//Create an S3Client
$s3 = new S3Client([
    'region' => 'eu-south-2'
]);
```

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 がデフォルトの認証情報プロバイダーチェーンを使用して認証情報を検索するためです。

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 のクライアントコンストラクタオプション」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と同じキー (「S3」、DynamoDb」など) を持つ関連付け配列で提供できます。

```
$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。

AWS SDK for PHP バージョン 3 のクライアントコンストラクタオプション

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。

クライアントに必要なクライアントコンストラクタオプションを明示的に指定しない場合、SDK for PHP は環境変数または AWS 設定ファイルから値を検索します。すべてのクライアントには認証情報プロバイダーの値と AWS リージョン 値が必要なため、これらの値をコンストラクタオプションまたは外部として設定する必要があります。

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 `credentials` ファイル`config`と ファイルの場所とフォーマットの詳細については、「 SDK とツールのリファレンスガイド」の「設定」を参照してください。 AWS SDKs

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);
```

クライアントの構築の詳細についてはthe section called “基本的なサービスクライアントの作成”、「」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 の認証情報ガイド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、ログに記録されたメッセージから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 と署名が に渡されますlogfn) 。

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)

再試行のレポートを無効にする`false`には、 に設定します。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)

`true` に設定すると、低レベルの HTTP アダプタからの統計情報 (例: GuzzleHttpTransferStats で返される値) の収集が有効になります。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リストについては、 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」および「region」キーを含むオプションのハッシュを受け入れる、省略可能な EndpointProviderV2 or PHP のインスタンスの呼び出し可能関数です。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-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' => 'amzn-s3-demo-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 = $client->getObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key'    => 'large.mov',
    '@http' => [
        'progress' => function ($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` オプションを使用して、プロキシを介して AWS サービスに接続できます。

- 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"`。
- 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、ホームディレクトリ (通常は) の認証情報ファイルから AWS 認証情報を作成するときに使用するプロファイルを指定します`~/.aws/credentials`。この設定は `AWS_PROFILE` 環境変数をオーバーライドします。

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

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

認証情報の設定と .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

AWS 接続先のリージョン。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

```
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 共有設定ファイル

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

```
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、 AWS リージョンを受け入れ、`Aws\Signature\SignatureInterface`オブジェクトを返すか、`NULL`プロバイダーが指定されたパラメータの署名者を作成できるかどうかを返す呼び出し可能な 。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

`[]`

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

`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

`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 AWS リージョン の の設定

SDK クライアントは、クライアントの作成時に AWS リージョン 指定した特定の AWS のサービスの に接続します。この設定により、アプリケーションはその地理的エリアの AWS リソースとやり

取りできます。リージョンを明示的に設定せずにサービスクライアントを作成すると、SDK は外部設定からデフォルトのリージョンを使用します。

リージョン解決チェーン

AWS SDK for PHP バージョン 3 では、次の順序を使用して、サービスクライアントが使用するリージョンを決定します。

1. コードで提供されるリージョン - クライアントコンストラクタオプションでリージョンを明示的に設定すると、他のすべてのソースよりも優先されます。

```
$s3Client = new Aws\S3\S3Client([
    'region' => 'us-west-2'
]);
```

2. 環境変数 — コードにリージョンが指定されていない場合、SDK はこれらの環境変数を順番にチェックします。
   - `AWS_REGION`
   - `AWS_DEFAULT_REGION`

```
# Example of setting Region through environment variables.
export AWS_REGION=us-east-1
```

3. AWS config files — リージョン環境変数が設定されていない場合、SDK は AWS 設定ファイルをチェックします。
   a. SDK は `~/.aws/config` (または`AWS_CONFIG_FILE`環境変数で指定された場所) を検索します。
   b. SDK は、`AWS_PROFILE`環境変数で指定されたプロファイル内のリージョン設定を調べます。
   c. `AWS_PROFILE` が指定されていない場合、SDK は「デフォルト」プロファイルを使用します。

   たとえば、次の設定ファイル設定があるとします。

```
# Example ~/.aws/config file.
[default]
region = eu-west-1

[profile production]
region = eu-central-1
```

`AWS_PROFILE` 環境変数が「本番稼働」の値で設定されている場合、クライアントは を使用します`eu-central-1 Region`。`AWS_PROFILE` 環境変数が存在しない場合、クライアントは `eu-west-1`リージョンを使用します。

4. 上記のソースのいずれかで SDK がリージョン値を見つけられない場合、リージョン値がサービスクライアントに必要な設定であるため、例外がスローされます。

ベストプラクティス

AWS SDK for PHP バージョン 3 でリージョンを使用する場合は、次のベストプラクティスを考慮してください。

本番コードでリージョンを明示的に設定する

本稼働アプリケーションでは、環境変数や に依存するのではなく、コードでリージョンを明示的に設定することをお勧めします`config`。これにより、コードの予測性が向上し、外部設定への依存が少なくなります。

開発とテストに環境変数を使用する

開発環境とテスト環境では、環境変数を使用すると、コードを変更せずに柔軟性が高まります。

複数の環境にプロファイルを使用する

アプリケーションで複数の AWS 環境を使用する必要がある場合は、 ファイルで異なるプロファイルを使用し、必要に応じてプロファイルを切り替えることを検討してください AWS `config`。

AWS SDK for PHP バージョン 3 認証情報プロバイダーの使用

AWS SDKs、「 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。 AWS SDKs

> **Important**
>
> セキュリティのため、ルートアカウントを AWS アクセスに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。

AWS SDK for PHP バージョン 3 の認証情報プロバイダーの役割は、認証情報を調達して SDK の AWS のサービス クライアントに提供することです。SDK は、ソースとなる認証情報を使用して、各リクエストに暗号で署名することでサービスで認証します。認証情報は通常、アクセスキー、つまりアクセスキー ID とシークレットアクセスキーで構成されます。

IAM Identity Center 認証を設定したり、IAM ロールを引き受けるようにランタイムを設定したりするなど、一時的な認証情報を使用すると、セッショントークンがアクセスキーに追加され、 AWS リソースへの時間制限付きアクセスが提供されます。

AWS SDK for PHP バージョン 3 の認証情報プロバイダーとは

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。SDK には、認証情報プロバイダー関数のいくつかの実装が用意されています。または、認証情報を作成したり、認証情報のロードを最適化したりするための独自のカスタムロジックを実装することもできます。

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

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

AWS SDK for PHP バージョン 3 のデフォルトの認証情報プロバイダーチェーンについて

デフォルトの認証情報プロバイダーチェーンは、SDK が呼び出す一連の組み込み認証情報プロバイダーで構成されます。これは、パラメータなしで defaultProvider 認証情報プロバイダー関数によって実装されます。有効な認証情報が見つかると、検索は停止されます。

は、認証情報プロバイダーを次の順序で AWS SDK for PHP 実行します。

- **`env` provider** - SDK AWS は、環境変数として設定されたアクセスキーを検索します。
- **`assumeRoleWithWebIdentityCredentialProvider`** provider - SDK は IAM ロールとウェブ ID トークンファイル設定を検索します。
- チェーンのこの時点で、SDK は共有 AWS `config` ファイルと `credentials` ファイルで設定を検索します。SDK は「デフォルト」プロファイルで設定を検索しますが、`AWS_PROFILE`環境変数が設定されている場合、SDK は名前付きプロファイル値を使用します。
  - **`sso`** provider - SDK は、共有`config`ファイル内の IAM Identity Center 設定を検索します。
  - **`process` provider** - SDK は共有`credentials`ファイルで `credential_process`設定を検索します。
  - **`ini`** provider - SDK は、共有`credentials`ファイル内の AWS 認証情報または IAM ロール情報を検索します。
  - **`process` provider** - SDK は共有`config`ファイルで `credential_process`設定を検索します。
  - **`ini`** provider - SDK は、共有`config`ファイル内の AWS 認証情報または IAM ロール情報を検索します。

- **`ecsCredentials`** provider - SDK は、一時的な認証情報を取得するための情報を提供する環境変数`AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` または `AWS_CONTAINER_CREDENTIALS_FULL_URI` を検索します。
- **`instanceProfile`** provider - SDK は EC2 インスタンスメタデータサービスを使用して、インスタンスプロファイルで指定された IAM ロールを取得します。ロール情報を使用して、SDK は一時的な認証情報を取得します。

> **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。

GitHub ソースコードでチェーンのコードを確認できます。

AWS SDK for PHP バージョン 3 の組み込み認証情報プロバイダー

SDK には、個別に使用したり、カスタム認証情報プロバイダーチェーンに結合したりできる組み込み認証情報プロバイダーがいくつか用意されています。

サービスクライアントの作成時に認証情報プロバイダーを指定すると、SDK は指定された認証情報プロバイダーのみを使用して認証情報のロードを試みます。デフォルトの認証情報プロバイダーチェーンは使用しません。サービスクライアントで`instanceProfile`プロバイダーを使用することがわかっている場合は、サービスクライアントコンストラクタで`instanceProfile`プロバイダーを指定することで、デフォルトのチェーンを短縮できます。

```
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',
    'credentials' => $memoizedProvider  // The default credential provider chain is not
 used.
]);
```

> **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。

トピック

- assumeRole SDK for PHP の プロバイダー
- sso SDK for PHP の プロバイダー
- defaultProvider SDK for PHP の プロバイダー

- ecsCredentials SDK for PHP の プロバイダー
- env SDK for PHP の プロバイダー
- assumeRoleWithWebIdentityCredentialProvider SDK for PHP の プロバイダー
- ini SDK for PHP の プロバイダー
- process SDK for PHP の プロバイダー
- instanceProfile SDK for PHP の プロバイダー

`assumeRole` SDK for PHP の プロバイダー

`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**
>
> すべての API オペレーションで AWS STS 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、認証情報の有効期限が切れたときに自動的に更新される認証情報を処理できます。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
]);
```

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

`sso` SDK for PHP の 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 = 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。

`defaultProvider` SDK for PHP の プロバイダー

`Aws\Credentials\CredentialProvider::defaultProvider` はデフォルトの認証情報プロバイダーであり、デフォルトの認証情報プロバイダーチェーンとも呼ばれます。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。たとえば、次のスニペットに示すように S3Client を作成すると、SDK はデフォルトのプロバイダーを使用します。

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

チェーン内の特定の認証情報プロバイダーにパラメータを指定する場合は、コードで defaultProvider を使用することもできます。たとえば、次の の例では、`ecsCredentials`プロバイダー関数が使用されている場合のカスタム接続タイムアウトと再試行設定を示します。

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

$provider = CredentialProvider::defaultProvider([
    'timeout' => '1.5',
    'retries' => 5
]);

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

`ecsCredentials` SDK for PHP の プロバイダー

`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 SDK for PHP の プロバイダー

環境変数を使用して認証情報を含めると、シー AWS クレットアクセスキーを誤って共有することを防ぐことができます。アクセス AWS キーを本稼働ファイル内のクライアントに直接追加しないことをお勧めします。

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、AWS SDK 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。

でアプリケーションをホストしている場合はAWS Elastic Beanstalk、、`AWS_ACCESS_KEY_ID`、および `AWS_SESSION_TOKEN`環境変数を AWS Elastic Beanstalk コンソールで設定してAWS_SECRET_KEY、SDK がそれらの認証情報を自動的に使用できるようにします。

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 SDKs」を参照してください。 https://docs.aws.amazon.com/sdkref/latest/guide/settings-reference.html#EVarSettings

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.
```

`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 SDK for PHP の 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、ロールを引き受けるために StsClient が使用する、設定されたリージョンを継承します。オプションで、完全な StsClient を提供できます。認証情報は、指定された StsClient として `false` で設定する必要があります。

```
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 SDK for PHP の プロバイダー

`Aws\Credentials\CredentialProvider::ini` は、共有 ファイル`config`と `credentials` ファイルから認証情報をロードしようとします。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`。SDK が`AWS_SDK_LOAD_NONDEFAULT_CONFIG`環境変数を検出すると、 にある共有 AWS `config`ファイル内の「デフォルト」プロファイルもチェックされます`~/.aws/config`。

```
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
]);
```

process SDK for PHP の プロバイダー

`Aws\Credentials\CredentialProvider::process` は、共有 AWS 設定ファイルのプロファイルで指定された`credential_process`値を実行して、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`ファイルから最初に「デフォルト」プロファイルをロードしようとします`~/.aws/credentials`。共有`credentials`ファイルに「デフォルト」プロファイルが見つからない場合、SDK は共有`config`ファイルでデフォルトプロファイルを検索します。以下は、共有`credentials`ファイルの設定例です。

```
[default]
credential_process = /path/to/file/credential_returning_executable.sh --custom-command
 custom_parameter
```

SDK は、PHP の `shell_exec`関数を使用して指定されたとおりに`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。は、次の形式で stdout に認証情報を書き込む`credential_process`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` SDK for PHP の プロバイダー

`Aws\Credentials\CredentialProvider::instanceProfile` は、Amazon EC2 インスタンスプロファイルで指定された IAM ロールの認証情報をロードしようとします。

```
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。再試行回数は `retries`オプションで設定でき、次のコード`0`に示すように オプションを 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);
```

環境変数AWS_METADATA_SERVICE_NUM_ATTEMPTSが使用可能な場合、その値は前述の「再試行」オプションよりも優先されます。

> **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。

SDK for PHP での認証情報プロバイダーの連鎖

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)
        )
    );
}
```

SDK for PHP で使用するカスタム認証情報プロバイダーの作成

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 && $secret) {
            return Create::promise_for(
                new Credentials($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));
    };
}
```

SDK for PHP での認証情報のメモリ

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。

AWS SDK for PHP バージョン 3 を使用して IAM ロールを引き受ける

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 ロールを使用して一時的なセキュリティ認証情報を取得 AWS 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と AWS SDK for PHP、コンソールによってインスタンスプロファイルが自動的に作成され、対応するロールと同じ名前が付けられます。

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、 AWS API コールを行う IAM ロールを引き受けることができます。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場合は、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。ロールを引き受けるには、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 ロールまたは AssumeRole の使用」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、 AWS STS クライアントを使用してユーザーを認証します。ID の webIdentityToken と ProviderId、およびユーザーのアクセス許可を持つ IAM ロールのロールの ARN を提供します。

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、 AWS SDK for PHP API リファレンスのAssumeRoleWithWebIdentity — ウェブベースの ID プロバイダーを介したフェデレーション」または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 ロールを使用する AWS SDK for PHP 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'
]);
```

SDK for PHP AWS STS で から一時的な認証情報を使用する

AWS Security Token Service（AWS STS) を使用すると、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特権、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、この`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は、 AWS STS 常に'Credentials'値が含まれます。を ($resultたとえば を使用してprint_r($result)) 出力すると、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。

SDK for PHP で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' => 'amzn-s3-demo-bucket',
    'Key'    => 'my-key',
]);
```

SDK for PHP で AWS 共通ランタイム (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 に対して優れたパフォーマンスと最小限のフットプリントで基本的な機能を提供します。このトピックでは、SDK for PHP AWS が CRT を使用するタイミングと、CRT AWS 拡張機能をインストールする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 ライブラリの認可と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。

AWS Common Runtime (AWS CRT) 拡張機能をインストールする

CRT 拡張機能をインストールする方法については、aws-crt-php AWS の GitHub リポジトリのメインページを参照してください。 GitHub aws-crt-php

AWS SDK for PHP バージョン 3 の使用

この章では、 AWS SDK for PHP バージョン 3 を使用して AWS のサービス を効果的に操作するための包括的なガイダンスを提供します。サービスリクエストの作成、エラーの処理、高度な SDK 機能を活用して で堅牢な PHP アプリケーションを構築するための重要なテクニックを学びます AWS。

シンプルなアプリケーションを構築する場合でも複雑なシステムを構築する場合でも、この章で説明する手法は、コードを最適化し、エラー処理を改善し、PHP SDK AWS のサービス を通じて を操作するための効率的なパターンを実装するのに役立ちます。

トピック

- AWS SDK for PHP バージョン 3 を使用した AWS のサービス リクエストの実行
- AWS SDK for PHP バージョン 3 を使用した非同期プログラミング
- AWS SDK for PHP バージョン 3 でのエラーの処理
- AWS SDK for PHP バージョン 3 のハンドラーとミドルウェア
- AWS SDK for PHP バージョン 3 のストリーム
- AWS SDK for PHP バージョン 3 でページ分割された結果を使用する
- AWS SDK for PHP バージョン 3 のウェーター
- AWS SDK for PHP バージョン 3 の JMESPath 式

AWS SDK for PHP バージョン 3 を使用した AWS のサービス リクエストの実行

SDK リクエストワークフローの概要

AWS SDK for PHP バージョン 3 の操作は、すべての で一貫したパターンに従います AWS のサービス。基本的なワークフローには、主に 3 つのステップがあります。

1. サービスクライアントを作成する — 使用する のクライアントオブジェクトをインスタンス化 AWS のサービス します。
2. オペレーションの実行 — サービスの API のオペレーションに対応するクライアントでメソッドを呼び出します。

3. 結果の処理 — 成功時に返された配列のような結果オブジェクトを使用するか、失敗時にスローされた例外を処理します。

以下のセクションでは、サービスクライアントを作成および設定する方法から始めて、これらの各ステップについて詳しく説明します。

基本的なサービス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 がデフォルトの認証情報プロバイダーチェーンを使用して認証情報を検索するためです。

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 のクライアントコンストラクタオプション」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。

上記の例は基本的なクライアント作成を示していますが、特定の要件を満たすようにサービスクライアントをカスタマイズできます。コードによるサービスクライアントの設定の詳細については、「」を参照してくださいAWS SDK for PHP バージョン 3 のコードでサービスクライアントを設定す

る。外部設定ファイルまたは環境変数を使用してサービスクライアントを設定する必要がある場合は、「」を参照してくださいAWS SDK for PHP バージョン 3 のサービスクライアントを外部で設定する。

リクエストの実行

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で、サービスリクエストを行うことができます。例えば、Amazon S3 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。クライアントを作成するときに、サービスモデルのversionパラメータ (2006-03-01 や最新など) を指定しない場合、クライアントはデフォルトで最新バージョンになります。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' => 'amzn-s3-demo-bucket',
    'Key'    => 'my-key',
    'Body'   => 'this is the body!',
    '@http'  => [
        'proxy' => 'http://192.168.16.1:10'
    ]
]);
```

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。

関連付け PHP 配列などのAWS\Result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');
```

AWS SDK for PHP バージョン 3 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、後で HTTP リクエストを転送するために使用されるパラメータ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 = 'amzn-s3-demo-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 を使用した非同期プログラミング

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();
    });
```

AWS SDK for PHP バージョン 3 の Promises

は promise AWS SDK for PHP を使用して非同期ワークフローを許可し、この非同期性により HTTP リクエストを同時に送信できます。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。

は promises 実装の guzzlehttp/promises Composer パッケージ AWS SDK for PHP に依存しています。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**
>
> HTTP リクエストは 1 つのスレッド AWS SDK for PHP を使用して で同時に送信されます。この場合、ノンブロッキング呼び出しを使用して 1 つ以上の HTTP リクエストを転送し、状態の変化 (例: promise の達成または拒否) に対応します。

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**
>
> の API オペレーション呼び出し AWS SDK for PHP 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 でのエラーの処理

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 のハンドラーとミドルウェア

を拡張する主なメカニズム AWS SDK for PHP は、ハンドラーとミドルウェアを使用することです。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');
```

ビルド

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'] = 'amzn-s3-demo-bucket';
        // 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' => 'amzn-s3-demo-bucket'
]);

// 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' => 'amzn-s3-demo-bucket'
```

```
]);

$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。

履歴

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 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、 は PHP ストリームに対する抽象化として内部的に PSR-7 StreamInterface AWS SDK for PHP を使用します。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 でページ分割された結果を使用する

一部の AWS サービスオペレーションはページ分割され、切り捨てられた結果で応答します。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、開発者 AWS SDK for PHP がページ分割された APIs。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' => 'amzn-s3-demo-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' => 'amzn-s3-demo-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'    => 'amzn-s3-demo-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' => 'amzn-s3-demo-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' => 'amzn-s3-demo-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 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' => 'amzn-s3-demo-bucket']);

// Wait until the created bucket is available
$s3Client->waitUntil('BucketExists', ['Bucket' => 'amzn-s3-demo-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'  => 'amzn-s3-demo-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'  => 'amzn-s3-demo-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' => 'amzn-s3-demo-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 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 には、バージョン 3 のページネーター AWS SDK for PHP やバージョン 3 のウェーターなどの高レベルの抽象化の一部を強化するために jmespath.php に依存していますが、 `Aws\ResultInterface` と で検索する JMESPath も公開しています`Aws\ResultPaginator`。 AWS SDK for PHP

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。上記と同じクエリを実行し、代わりにボリュームが「us-」で始まる AWS リージョンにあるすべてのボリュームを取得するとします。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 を使用すると、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て反復処理できます。基本的に、JMESPath 式の結果がマップ関数であるイテレーターにフラットマップを実装します。

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 SDK for PHP バージョン 3 AWS のサービス からの呼び出し

以下のセクションには、 を使用して AWS サービス AWS SDK for PHP 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
- のガイダンスを含むコード例 AWS SDK for PHP

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 のサービス APIs。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
- AWS SDK for PHP バージョン 3 の Amazon S3 の機能とオプション

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AWS AWS 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();
```

設定

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。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。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。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。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地理的に最も近い AWS リージョンまたは同じ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 を使用して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需要に合わせて必要に応じて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。

AWS SDK for PHP バージョン 3 の Amazon S3 の機能とオプション

このトピックでは、Amazon S3 と連携するために AWS SDK for PHP バージョン 3 が提供する追加機能とオプションについて説明します。 Amazon S3

トピック

- AWS SDK for PHP バージョン 3 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 の Amazon S3 ストリームラッパー
- AWS SDK for PHP バージョン 3 の Amazon S3 Transfer Manager
- AWS SDK for PHP バージョン 3 での Amazon S3 クライアント側の暗号化
- チェックサムによるデータ整合性保護

AWS SDK for PHP バージョン 3 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 には、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用意されています。これにより、ユーザーは任意のコマンドに入力`@region`パラメータを指定して、コマンドを送信する AWS リージョンを指定できます。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。これは、バケットが複数のリージョンに存在するAWS SDK for PHP バージョン 3 の Amazon S3 ストリームラッパーのユーザーにとって特に便利な機能です。

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唯一の使用上の違いは、`@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

Amazon S3 マルチリージョンクライアントは、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を維持します。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 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://amzn-s3-demo-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 What's New with AWS 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://amzn-s3-demo-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://amzn-s3-demo-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「amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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 の Amazon S3 Transfer Manager

の Amazon S3 転送マネージャーは、ディレクトリ全体を Amazon S3 バケットにアップロードし、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 AWS SDK for PHP 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();
```

設定

`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 での 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。

はエンベロープ暗号化 AWS SDK for PHP 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。

セットアップ

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 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。

Encryption

`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**
>
> `'@CipherOptions'` が正しく設定されていない場合、Amazon S3 および AWS KMSベースのサービスエラーに加えて、スローされた`InvalidArgumentException`オブジェクトが表示されることがあります。

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**
>
> `'@CipherOptions'` が正しく設定されていない場合、Amazon S3 および AWS KMSベースのサービスエラーに加えて、スローされた`InvalidArgumentException`オブジェクトが表示されることがあります。

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 SDKs でサポートされていないため、他の SDKs 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できない場合があります。

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**
>
> `'@CipherOptions'` が正しく設定されていない場合、Amazon S3 および AWS KMSベースのサービスエラーに加えて、スローされた`InvalidArgumentException`オブジェクトが表示されることがあります。

チェックサムによるデータ整合性保護

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 のチェックサムの詳細については、サポートされているアルゴリズムを含む Amazon Simple Storage Service ユーザーガイドを参照してください。

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アルゴリズムのいずれかを使用して、事前に計算されたチェックサム値を指定することもできます。

> **Note**
>
> SDK には、外部で設定できるデータ整合性保護のグローバル設定も用意されています。詳細については、AWS SDKs」を参照してください。

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

リクエストでチェックサムアルゴリズムを指定しない場合、チェックサムの動作は、次の表に示すように、使用する SDK のバージョンによって異なります。

チェックサムアルゴリズムが指定されていない場合のチェックサム動作

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、SHA256 チェックサムが事前に計算された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

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。

> **Note**
> 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。

のガイダンスを含むコード例 AWS SDK for PHP

このセクションでは、 を使用する一般的な AWS シナリオを示すコード例を示します AWS SDK for PHP。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

トピック

- AWS SDK for PHP バージョン 3 を使用した Amazon CloudFront の例
- 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 の例
- AWS SDK for PHP バージョン 3 で Amazon OpenSearch Service 検索リクエストに署名する
- AWS Identity and Access ManagementAWS SDK for PHP バージョン 3 を使用した の例
- AWS Key Management ServiceAWS 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 Kinesis の例
- 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 と 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 の例
- AWS SDK for PHP バージョン 3 を使用した Amazon SNS の例
- AWS SDK for PHP バージョン 3 を使用した Amazon SQS の例
- AWS SDK for PHP バージョン 3 を使用して 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 CloudFront の例

Amazon CloudFront は、独自の AWS ウェブサーバーまたは Amazon S3 などの AWS サーバーからの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 のすべてのサンプルコードは、GitHub で入手できます。

Amazon 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 CloudFront は、世界各地にあるエッジロケーションのコンテンツをキャッシュして、お客様独自のサーバーまたは Amazon S3 や Amazon EC2 などの Amazon のサービスに保存する静的ファイルおよび動的ファイルの配信を高速化します。ユーザーがお客様のウェブサイトのコンテンツをリクエストすると、CloudFront は、最も近いエッジロケーションからそのコンテンツを配信します (ファイルがそこにキャッシュされている場合)。それ以外の場合、CloudFront はファイルのコピーを取得し、配信してから、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したときの配信方法を指定します。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で

CloudFront を使用する方法の詳細については、「CloudFront 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」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。

- CreateDistribution を使用してディストリビューションを作成します。
- GetDistribution を使用してディストリビューションを取得します。
- ListDistributions を使用してディストリビューションをリストします。
- UpdateDistributions を使用してディストリビューションを更新します。
- DisableDistribution を使用してディストリビューションを無効にします。
- DeleteDistributions を使用してディストリビューションを削除します。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

Amazon CloudFront の使用に関する詳細については、「Amazon CloudFront デベロッパーガイド」を参照してください。

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の取得

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のリスト

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された AWS リージョン内の既存の 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の更新

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類似しています。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の無効化

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の削除

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。

指定された 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();
```

Amazon CloudFront API と AWS SDK for PHP バージョン 3 を使用した Amazon CloudFront 無効化の管理

Amazon CloudFront は、世界中のエッジロケーションで静的ファイルおよび動的ファイルのコピーをキャッシュします。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のコンテンツの削除の詳細については、「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。

- CreateInvalidation 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。

- GetInvalidation 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。
- ListInvalidations を使用してディストリビューションをリストします。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

Amazon CloudFront の使用に関する詳細については、「Amazon CloudFront デベロッパーガイド」を参照してください。

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位置を指定して、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

現在のすべての CloudFront ディストリビューションの無効化をリストするには、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セットアップする方法および URL の署名が必要な理由の詳細については、Amazon CloudFront デベロッパーガイドの「Amazon CloudFron を使用したプライベートコンテンツの提供」を参照してください。

- getSignedURL を使用して、署名付きの Amazon CloudFront を作成します。
- getSignedCookie を使用して、署名付きの Amazon CloudFront Cookie を作成します。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

Amazon CloudFront の使用に関する詳細については、「Amazon CloudFront デベロッパーガイド」を参照してください。

プライベートディストリビューションの CloudFront URL 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。既定またはカスタムポリシーを使用して、ビデオリソースの CloudFront URL に署名することができます。

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();
```

CloudFront URL 作成時のカスタムポリシーの使用

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 の使用

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。同様の手法が FlowPlayer などの他のプレイヤーに適用されますが、異なるクライアント側コードが必要になります。

```
<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 URL の代わりに、署名付き Cookie が必要になる (またはその逆の) 理由の詳細については、Amazon CloudFront デベロッパーガイドの「署名付き URL と署名付き Cookie の選択」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 作成時のカスタムポリシーの使用

`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 クライアントへの CloudFront Cookie の送信

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の署名

Amazon CloudSearch ドメインリクエストは、 AWS SDK for PHPでサポートされているもの以外でもカスタマイズできます。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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

Amazon CloudSearch ドメインリクエストに署名

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 を使用してメトリクスを収集および追跡できます。メトリクスとは、リソースやアプリケーションについて測定できる変数です。CloudWatch アラームは、ユーザーが定義したルールに基づいて、通知を送信したり、モニタリングしているリソースに自動的に変更を加えたりします。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 を使用した Amazon CloudWatch からのメトリクスの取得
- AWS SDK for PHP バージョン 3 を使用した Amazon CloudWatch でのカスタムメトリクスの発行
- 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 CloudWatch アラームでの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。

- DescribeAlarms を使用したアラームの記述。
- PutMetricAlarm を使用したアラームの作成。

- DeleteAlarms を使用したアラームの削除。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 を使用した Amazon CloudWatch 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。

- ListMetrics を使用して、メトリクスのリストを取得します。
- DescribeAlarmsForMetric を使用して、メトリクスに対するアラームを取得します。
- GetMetricStatistics を使用して、特定のメトリクスの統計情報を取得します。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 を使用した Amazon CloudWatch でのカスタムメトリクスの発行

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。

- PutMetricData を使用したメトリクスデータの発行。
- PutMetricAlarm を使用したアラームの作成。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 Events では、Amazon Web Services (AWS) リソースの変更を記述した、システムイベントのほぼリアルタイムのストリーミング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。

- PutRule を使用してルールを作成します。
- PutTargets を使用してルールにターゲットを追加します。
- PutEvents を使用した CloudWatch Events へのカスタムイベントの送信

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 CloudWatch アラームでの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。

- EnableAlarmActions 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。
- DisableAlarmActions 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。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 の例

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。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 での 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 での Amazon EC2 キーペアの使用
- AWS SDK for PHP バージョン 3 での Amazon EC2 のセキュリティグループの使用

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。

- DescribeInstances を使用して Amazon EC2 インスタンスを記述します。
- MonitorInstances 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。
- UnmonitorInstances 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。
- StartInstances を使用して以前に停止した Amazon EBS-backed AMI を開始します。
- StopInstances を使用して Amazon EBS-backed インスタンスを停止します。
- RebootInstances 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。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 での 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 に関連付けられています AWS アカウント。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。

- DescribeInstances を使用して、1 つ以上のインスタンスを記述します。
- AllocateAddress を使用して、Elastic IP アドレスを取得します。
- AssociateAddress 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。
- ReleaseAddress を使用して、Elastic IP アドレスを解放します。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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で構成されます。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。

- DescribeAvailabilityZones を使用して、ユーザーが利用可能なアベイラビリティーゾーンを記述します。
- AWS DescribeRegions を使用して、現在利用可能なリージョンについて説明します。DescribeRegions

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 での Amazon EC2 キーペアの使用

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。

- CreateKeyPair を使用した 2048 ビットの RSA キーペアの作成。
- DeleteKeyPair を使用した、指定キーペアの削除。
- DescribeKeyPairs を使用した 1 つ以上のキーペアの記述。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 での Amazon EC2 のセキュリティグループの使用

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。

- DescribeSecurityGroups 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。
- AuthorizeSecurityGroupIngress 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。
- CreateSecurityGroup を使用して、セキュリティグループを作成します。
- DeleteSecurityGroup を使用して、セキュリティグループを削除します。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 で Amazon OpenSearch Service 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。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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、サービスの署名名 (esこの場合は) と OpenSearch Service ドメインの AWS リージョンに対して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完全なリストは、 のAWS「リージョンとエンドポイント」ページにあります Amazon Web Services 全般のリファレンス。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。

認証情報を指定する必要があります。これは、SDK のデフォルトのプロバイダーチェーンまたはAWS SDK for PHP バージョン 3 の認証情報で説明されている任意の形式の認証情報で実行できます。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 ManagementAWS SDK for PHP バージョン 3 を使用した 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、 AWS 製品を使用するクラウド内の複数のユーザーまたはシステムを持つ組織を対象としています。IAM を使用すると、ユーザー、アクセスキーなどのセキュリティ認証情報、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アクセス許可を一元管理できます。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 IAM 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 IAM 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。

- CreateAccessKey を使用した、シークレットアクセスキーと対応するアクセスキー ID の作成。
- ListAccessKeys を使用した、IAM ユーザーに関連付けられたアクセスキー ID に関する情報の取得。
- GetAccessKeyLastUsed を使用した、アクセスキーの最終使用日時に関する情報の取得。
- UpdateAccessKey を使用した、アクセスキーのステータスの変更 (アクティブから非アクティブまたはその逆)。
- DeleteAccessKey を使用した、IAM ユーザーに関連付けられたアクセスキーペアの削除。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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、 で作成するエンティティで、それを使用してやり取りするユーザーまたはサービス AWS を表します AWS。のユーザーは、名前と認証情報 AWS で構成されます。

以下の例では、次の方法を示しています。

- CreateUser を使用して新しい IAM ユーザーを作成する
- ListUsers を使用して IAM ユーザーのリストを取得する
- UpdateUser を使用して IAM ユーザーを更新する
- GetUser を使用して IAM ユーザーに関する情報を取得する
- DeleteUser を使用して IAM ユーザーを削除する

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 IAM 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 の代わりに会社名やその他のわかりやすい識別子を含める場合は、 AWS アカウント ID のエイリアスを作成できます。 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 が変更され、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。

- CreateAccountAlias を使用してエイリアスを作成します。
- ListAccountAliases 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。
- DeleteAccountAlias を使用してエイリアスを削除します。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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。

- CreatePolicy を使用して管理ポリシーを作成する
- AttachRolePolicy を使用してポリシーをロールにアタッチする
- AttachUserPolicy を使用してポリシーをユーザーにアタッチする
- AttachGroupPolicy を使用してポリシーをグループにアタッチする
- DetachRolePolicy を使用してロールポリシーを削除する
- DetachUserPolicy を使用してユーザーポリシーを削除する
- DetachGroupPolicy を使用してグループポリシーを削除する
- DeletePolicy を使用して管理ポリシーを削除する
- DeleteRolePolicy を使用してロールポリシーを削除する
- DeleteUserPolicy を使用してユーザーポリシーを削除する
- DeleteGroupPolicy を使用してグループポリシーを削除する

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 IAM サーバー証明書の使用

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 AWS、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Webサイトまたはアプリケーションで AWSに使用するには、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。

- ListServerCertificates を使用して、IAM に保存されている証明書のリストを取得します。
- GetServerCertificate を使用して証明書に関する情報を取得する
- UpdateServerCertificate を使用して証明書を更新する
- DeleteServerCertificate を使用して証明書を削除する

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 ServiceAWS SDK for PHP バージョン 3 を使用した 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制御を容易にするマネージドサービスです。詳細については AWS KMS、Amazon KMS ドキュメントを参照してください。安全な PHP アプリケーションを記述する場合も、他の AWS サービスにデータを送信する場合 AWS KMS も、キーを使用できるユーザーと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 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 と 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と復号
- 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 と AWS SDK for PHP バージョン 3 を使用した許可の使用
- AWS KMS API と 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 と 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) のプライマリリソースは ですAWS KMS keys。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。

- CreateKey を使用してカスタマー KMS キーを作成する。

- GenerateDataKey を使用してデータキーを生成する。
- DescribeKey を使用して KMS キーを表示する。
- ListKeys を使用して KMS キーのキー ID とキー ARN を取得する
- EnableKey を使用して KMS キーを有効にする。
- DisableKey を使用して KMS キーを無効にする

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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) およびキーの状態を含む KMS キーについての詳細を取得するには、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

無効になっている 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と復号

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の (AWS KMS) を使用して、データキーAWS KMS key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。
- ReEncrypt 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。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 `Encrypt` 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。

指定する `ciphertextBlob` は、GenerateDataKey、GenerateDataKeyWithoutPlaintext、または暗号化レスポンスからの `CiphertextBlob` フィールド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、続いてすぐに異なる KMS キーでデータキーを再暗号化するには、ReEncrypt オペレーションを使用します。オペレーションは内部のサーバー側で完全に実行されるため AWS KMS、プレーンテキストが外部に公開されることはありません AWS KMS。

指定する ciphertextBlob は、GenerateDataKey、GenerateDataKeyWithoutPlaintext、または暗号化レスポンスからの CiphertextBlob フィールド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。キーポリシーを使用して、カスタマーマネージド KMS キーのアクセス許可をいつでも追加、削除、または変更できますが、 AWS マネージド 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。

- ListKeyPolicies を使用してキーポリシーの名前をリストする。
- GetKeyPolicy を使用してキーポリシーを取得する。
- PutKeyPolicy を使用してキーポリシーを設定する。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 と AWS SDK for PHP バージョン 3 を使用した許可の使用

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。権限を使用して、プリン AWS シパルが AWS Key Management Service（AWS KMS) カスタマー管理のを使用できるようにする長期的なアクセスを許可できます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。

- CreateGrant を使用して KMS キーに対するグラントを作成する。
- ListGrants を使用して KMS キーの許可を表示する。
- RetireGrant を使用して KMS キーの許可を無効にする。
- RevokeGrant を使用して KMS キーの許可を取り消す。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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

の許可を廃止するには 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 と 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AWS KMS keyと呼ばれる のオプションの表示名を提供します。

以下の例では、次の方法を示しています。

- CreateAlias を使用してエイリアスを作成する。
- ListAliases を使用してエイリアスを表示する。
- UpdateAlias を使用してエイリアスを更新する。
- DeleteAlias を使用してエイリアスを削除する。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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必要があります。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

発信者の 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 で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、または 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、GitHub で入手できます。

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 を使用してデータシャードを管理する
- 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。

- CreateAlias を使用してデータストリームを作成する。
- DescribeStream を使用して 1 つのデータストリームの詳細を取得する。
- ListStreams を使用して既存のデータストリームをリストする。

- PutRecord を使用して既存のデータストリームにデータを送信する。
- DeleteStream を使用してデータストリームを削除する。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

「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。

指定した Kinesis データストリームに関する詳細情報を取得するには、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 からの最初の 10 個のデータストリーム AWS アカウント を一覧表示します。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リストするには、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。最大 500 個のレコードを 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。

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 を使用してデータシャードを管理する

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。

- ListShards を使用してストリーム内のシャードをリストする。
- UpdateShardCount を使用してストリーム内のシャードの数を追加または削減する。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 データストリームにあるシャードをリストするには、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 Service)、Amazon Redshift などの他の AWS 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。

- CreateDeliveryStream を使用して配信ストリームを作成する。
- DescribeDeliveryStream を使用して 1 つの配信ストリームの詳細を取得する
- ListDeliveryStreams を使用して配信ストリームをリストする。
- PutRecord を使用して配信ストリームにデータを送信する。
- DeleteDeliveryStream を使用して配信ストリームを削除する。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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、開発者は既存の 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 S3 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、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。次に、Amazon S3送信先への Kinesis Data Firehose アクセスの付与」で説明されているように、Amazon S3 バケットへの Firehose アクセスを許可してください。

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 Service を使用して配信ストリームをテストする

OpenSearch Service 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 Kinesis Data Firehose アクセスの付与」の説明に従って、OpenSearch Service クラスターへの Firehose アクセスを許可してください。 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すべての Firehose 配信ストリームを一覧表示するには、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する既存の配信ストリームを一覧表示する

Amazon S3、OpenSearch Service、または Amazon Redshift、または Splunk にデータを送信する既存のすべての Firehose 配信ストリームを一覧表示するには、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宛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ディングサービスです。このサービスでは、インターネット全体に配信するブロードキャストおよびビデオオンデマンド (VOD) 用の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 、トランスコーディング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
- 完了したコード変換ジョブの JSON を取得します。GetJob
- 最近作成されたジョブの最大 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 を使用した Amazon S3 バケットのアクセス許可の管理
- AWS SDK for PHP バージョン 3 を使用した 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 を使用した Amazon S3 の署名付き POSTs
- 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 ARNs 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。

- ListBuckets を使用した、リクエストの認証送信者が所有するバケットリストの取得。

- CreateBucket を使用した、新しいバケットの作成。
- PutObject を使用した、バケットへのオブジェクトの追加。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 を使用した Amazon S3 バケット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。ACLs 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。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 = 'amzn-s3-demo-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 を使用した 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。

- GetBucketCors を使用して、バケットの CORS 設定を取得する
- PutBucketCors を使用して、バケットの CORS 設定を設定する

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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。

- ObjectUploader を使用して、Amazon S3 にオブジェクトをアップロードする。
- MultipartUploader 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する。
- ObjectCopier を使用して、Amazon S3 のある場所から別の場所にオブジェクトをコピーする。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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わからない場合は、 を使用しますObjectUploader。 は、ペイロードサイズに基づいてMultipartUploader、 PutObjectまたは を使用して大きなファイルを Amazon S3 ObjectUploader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);
```

設定

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。CommandInterface オブジェクトに追加できるパラメータについては、CompleteMultipartUpload API リファレンスを参照してください。

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。ファイルサイズが mup_threshold値を超えた場合、SDK はこのコールバックを呼び出します。CommandInterface オブジェクトに追加できるパラメータについては、CreateMultipartUpload API リファレンスを参照してください。

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。ファイルサイズが mup_threshold値以下の場合、SDK はこのコールバックを呼び出します。PutObject リクエストに適用できるパラメータについては、PutObject API リファレンスを参照してください。UploadPart リクエストに適用されるパラメータについては、UploadPart API リファレンスを参照してください。SDK は、 CommandInterface オブジェクトで表されるオペレーションに適用されないパラメータを無視します。

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。

params

各サブコマンドの設定オプションを提供する連想配列。例:

```
new ObjectUploader($bucket, $key, $body, $acl, ['params' => ['CacheControl'
 => <some_value>])
```

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();
```

設定

`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。

params

各サブコマンドの設定オプションを提供する連想配列。例:

```
new MultipartUploader($client, $source, ['params' => ['CacheControl'
 => <some_value>]])
```

マルチパートコピー

には、 と同様の方法で使用されるMultipartCopyオブジェクト AWS SDK for PHP も含まれていますがMultipartUploader、Amazon S3 内で 5 GB ~ 5 TB のサイズのオブジェクトをコピーするように設計されています。

```
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。「署名付き」リクエストを作成し、別のユーザーが使用できる URL としてエンコードするという考えです。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。

HTTP GET リクエストの署名付き URL を作成する

次のコード例は、 SDK for PHP を使用して HTTP GET リクエストの署名付き URL を作成する方法を示しています。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

$s3Client = new S3Client([
    'region' => 'us-west-2',
]);

// Supply a CommandInterface object and an expires parameter to the
 `createPresignedRequest` method.
$request = $s3Client->createPresignedRequest(
    $s3Client->getCommand('GetObject', [
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key' => 'demo-key',
    ]),
    '+1 hour'
);

// From the resulting RequestInterface object, you can get the URL.
$presignedUrl = (string) $request->getUri();
```

```
echo $presignedUrl;
```

メソッドの API リファレンスcreatePresignedRequestには、詳細が記載されています。

他のユーザーは $presignedUrl値を使用して、次の 1 時間以内にオブジェクトを取得できます。例えば、ブラウザを使用して HTTP GET リクエストが行われると、S3 サービスには、通話が署名付き URL を作成したユーザーからのものであることが表示されます。

HTTP PUT リクエストの署名付き URL を作成する

次のコード例は、SDK for PHP を使用して HTTP PUT リクエストの署名付き URL を作成する方法を示しています。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

$s3Client = new S3Client([
    'region' => 'us-west-2',
]);

$request = $s3Client->createPresignedRequest(
    $s3Client->getCommand('PutObject', [
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key' => 'demo-key',
    ]),
    '+1 hour'
);

// From the resulting RequestInterface object, you can get the URL.
$presignedUrl = (string) $request->getUri();
```

他のユーザーが HTTP PUT リクエストで署名付き URL を使用してファイルをアップロードできるようになりました。

```
use GuzzleHttp\Psr7\Request;
use GuzzleHttp\Psr7\Response;

// ...
```

```
function uploadWithPresignedUrl($presignedUrl, $filePath, $s3Client): ?Response
{
    // Get the HTTP handler from the S3 client.
    $handler = $s3Client->getHandlerList()->resolve();

    // Create a stream from the file.
    $fileStream = new Stream(fopen($filePath, 'r'));

    // Create the request.
    $request = new Request(
        'PUT',
        $presignedUrl,
        [
            'Content-Type' => mime_content_type($filePath),
            'Content-Length' => filesize($filePath)
        ],
        $fileStream
    );

    // Send the request using the handler.
    try {
        $promise = $handler($request, []);
        $response = $promise->wait();
        return $response;
    } catch (Exception $e) {
        echo "Error uploading file: " . $e->getMessage() . "\n";
        return null;
    }
}
```

AWS SDK for PHP バージョン 3 を使用した Amazon S3 の署名付き POSTs

署名付き URLs と同様に、署名付き POSTs を使用すると、 AWS 認証情報を付与せずにユーザーに書き込みアクセス権を付与できます。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。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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
- 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 = 'amzn-s3-demo-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。

- GetBucketWebsite を使用した、バケットのウェブサイト設定の取得。

- PutBucketWebsite を使用した、バケットのウェブサイト設定のセットアップ。
- DeleteBucketWebsite を使用した、バケットのウェブサイト設定の削除。

AWS SDK for PHP バージョン 3 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 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。

- GetBucketPolicy を使用した、指定バケットのポリシーの取得。

- PutBucketPolicy を使用した、バケットでのポリシーの置き換え。
- DeleteBucketPolicy を使用したバケットからのポリシーの削除。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 ARNs 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代わりに、バケットフィールドのアクセスポイント ARNs 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。

- GetObject をアクセスポイントの ARN とともに使用して、バケットからオブジェクトを取得します。
- バケットにオブジェクトを追加するには、アクセスポイントの ARN とともに PutObject を使用します。

- 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 AWS 共有設定ファイルの config 変数 (デフォルトでは)~/.aws/config。

```
[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、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 CLI を使用して作成できます。 AWS S3 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、 メソッド`getObject()`と `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 と 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シークレットを保存および管理します。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。Secrets Manager は、 を使用してサービス固有の詳細を実装することで、他のデータベースやサードパーティーサービスのシークレット AWS Lambda をシームレスにローテーションすることもできます。

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。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 この例では、 AWS Secrets Manager API リファレンスから次の REST オペレーションを使用します。

- CreateSecret
- GetSecretValue
- ListSecrets
- DescribeSecret
- PutSecretValue
- RotateSecret
- DeleteSecret

の使用の詳細については AWS Secrets Manager、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 の例

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 は Amazon SES サービスの 2 つのバージョンを提供し、それに応じて SDK for PHP は SesClient と SesV2Client の 2 つのバージョンのクライアントを提供します。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 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 と AWS SDK for PHP バージョン 3 を使用した E メール ID の検証
- Amazon SES API とバージョン 3 を使用したカスタム E メールテンプレートの作成 AWS SDK for PHP
- Amazon SES API とバージョン 3 を使用した E メールフィルターの管理 AWS SDK for PHP
- Amazon SES API とバージョン 3 を使用した E メールルールの作成と管理 AWS SDK for PHP
- Amazon SES API と AWS SDK for PHP バージョン 3 を使用した送信アクティビティのモニタリング
- Amazon SES API と AWS SDK for PHP バージョン 3 を使用した送信者の承認

Amazon SES API と AWS SDK for PHP バージョン 3 を使用した E メール ID の検証

Amazon Simple Email Service (Amazon SES) アカウントの使用を初めて開始するときは、E メールを送信するのと同じ AWS リージョンですべての送信者と受信者を検証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。

- VerifyEmailIdentity を使用して E メールアドレスを確認する。
- VerifyDomainIdentity を使用して E メールドメインを確認する。
- ListIdentities を使用して E メールアドレスをリストする。
- ListIdentities を使用して E メールドメインをリストする。
- DeleteIdentity を使用して E メールアドレスを削除する。
- DeleteIdentity を使用して E メールドメインを削除する。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 アカウントに送信側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 のリストから確認済み 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

確認済み ID のリストから確認済み 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 とバージョン 3 を使用したカスタム E メールテンプレートの作成 AWS SDK for PHP

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。

- CreateTemplate を使用して E メールテンプレートを作成する。
- ListTemplates を使用してすべての E メールテンプレートをリストする。
- GetTemplate を使用して E メールテンプレートを取得する。
- UpdateTemplate を使用して E メールテンプレートを更新する。
- DeleteTemplate を使用して E メールテンプレートを削除する。

- SendTemplatedEmail を使用してテンプレート E メールを送信する。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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ための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。テンプレートは、テンプレートが追加された AWS リージョンでメッセージを送信する権限を持つ任意のアカウントで使用できます。

> **Note**
>
> Amazon SES は HTML を検証しないため、E メールを送信する前に必ず 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コンテンツを表示するには、GetTemplate オペレーションを使用します。TemplateName のみ必須です。

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、プレーンテキストを含む特定の E メールテンプレートのコンテンツを変更するには、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 TemplateName だけです。

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->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 E メールを受取人に送信するには、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 とバージョン 3 を使用した E メールフィルターの管理 AWS SDK for PHP

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。

- CreateReceiptFilter を使用して E メールフィルターを作成する。
- ListReceiptFilters を使用してすべての E メールフィルターをリストする。
- DeleteReceiptFilter を使用して E メールフィルターを削除する。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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拒否するには、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 とバージョン 3 を使用した E メールルールの作成と管理 AWS SDK for PHP

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、Amazon S3、Amazon SNS Amazon SNS 、 などの他の AWS サービスに E メールを送信できます AWS Lambda。

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。

- CreateReceiptRuleSet を使用して受信ルールセットを作成する。
- CreateReceiptRule を使用して受信ルールを作成する。
- DescribeReceiptRuleSet を使用して受信ルールセットを記述する。
- DescribeReceiptRule を使用して受信ルールを記述する。
- ListReceiptRuleSets を使用してすべての受信ルールセットをリストする。
- UpdateReceiptRule を使用して受信ルールを更新する。
- DeleteReceiptRule を使用して受信ルールを削除する。

- DeleteReceiptRuleSet を使用して受信ルールセットを削除する。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 RuleSetName を指定し、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、ルールと RuleSetName を CreateReceiptRule オペレーションに指定します。

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、RuleSetName 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->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 RuleSetName 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->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

この例では、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 RuleSetName 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';
$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、RuleSetName を DeleteReceiptRuleSet オペレーションに設定します。

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、RuleName と RuleSetName を DeleteReceiptRule オペレーションに設定します。

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 を使用した送信アクティビティのモニタリング

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。

- GetSendQuota を使用して送信クォータを確認する。
- GetSendStatistics を使用して送信アクティビティをモニタリングする。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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。送信可能な残りのメッセージ数を確認するには、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 週間以内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 と AWS SDK for PHP バージョン 3 を使用した送信者の承認

ユーザーに代わって AWS アカウント別の、 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、送信承認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。

- PutIdentityPolicy を使用して承認済み送信者を作成する。
- GetIdentityPolicies を使用して承認済み送信者のポリシーを取得する。
- ListIdentityPolicies を使用して承認済み送信者をリストする。
- DeleteIdentityPolicy を使用して承認済み送信者のアクセス許可を取り消す。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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

ユーザーに代わって E メールを送信 AWS アカウント することを他のユーザーに許可するには、ID ポリシーを使用して、検証済みの 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。アイデンティティ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。指定された 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 オペレーションで関連付けられた 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 するための別の 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 の例

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するときに、サポートされているプロトコル (Amazon SQS、HTTP/HTTPS URLs、E メール AWS SMS、Lambda) のいずれかを介してメッセージまたは通知を消費または受信します。

AWS SDK for PHP バージョン 3 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 を使用して Amazon SNS で SMS メッセージを送信する

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 URLs、E メール AWS SMS、または に通知を送信するには AWS Lambda、まず、そのトピックの受信者へのメッセージの配信を管理するトピックを作成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。

- CreateTopic を使用して通知を発行するトピックを作成する。
- ListTopics を使用してリクエスタのトピックのリストを返す。
- DeleteTopic 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する。
- GetTopicAttributes を使用してトピックのすべてのプロパティを返す。

- SetTopicAttributes を使用して、トピックの所有者がトピックの属性を新しい値に設定できるようにする。

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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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。

の各トピック名は一意 AWS アカウント である必要があります。

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

1 つ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

1 つ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 に通知を送信します AWS Lambda。

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。
- ConfirmSubscription を使用してサブスクリプションを確認する。
- ListSubscriptionsByTopic を使用して既存のサブスクリプションをリストする。
- 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 個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の削除に認証が必要な場合は、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がサブスクリプションを解除でき、 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 を使用して Amazon SNS で SMS メッセージを送信する

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。
- CheckIfPhoneNumberISOptedOut を使用して、指定された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ているかどうかを調べる。
- CheckIfPhoneNumberISOptedOut を使用して、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ている電話番号をリストする。
- 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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

指定された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ているかどうかを調べるには、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ている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 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 で Amazon SQS でロングポーリングを有効にする
- AWS SDK for PHP バージョン 3 を使用した 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 での 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 での Amazon SQS でのキューの使用

AWS SDK for PHP バージョン 3 で Amazon SQS でロングポーリングを有効にする

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。

- SetQueueAttributes を使用して、Amazon SQS キューの属性を設定し、ロングポーリングを有効にします。
- ReceiveMessage 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する
- CreateQueue を使用して、ロングポーリングキューを作成する

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 を使用した 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。

- ChangeMessageVisibilityBatch を使用した、キュー内指定メッセージの可視性タイムアウトの新しい値への変更。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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。

- SendMessage を使用した、指定キューへのメッセージの配信。
- ReceiveMessage を使用した、指定キューからのメッセージ (1 から 10 個) の取得。
- DeleteMessage を使用した、キューからのメッセージの削除。

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 での 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。

- SetQueueAttributes を使用してデッドレターキューを有効にする

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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 での 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。

- ListQueues を使用してキューのリストを取得する

- CreateQueue を使用して新しいキューを作成する
- GetQueueUrl を使用して既存のキューの URL を取得する
- DeleteQueue を使用して、指定したキューを削除する

のすべてのサンプルコード AWS SDK for PHP 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、「」の説明に従って AWS 認証情報を設定しますthe section called “による認証 AWS”。次に AWS SDK for PHP、「」の説明に従って をインポートしますthe section called “インストール”。

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());
}
```

AWS SDK for PHP バージョン 3 を使用して 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`、EventBridge グローバルエンドポイントに単一のイベントを送信します。

```
<?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 for PHP コードサンプル

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 の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。

一部のサービスには、サービス固有のライブラリや関数の活用方法を示す追加のカテゴリ例が含まれています。

サービス

- SDK for PHP を使用した API Gateway の例
- SDK for PHP を使用した Aurora の例
- SDK for PHP を使用した自動スケーリングの例
- SDK for PHP を使用した Amazon Bedrock の例
- SDK for PHP を使用した Amazon Bedrock ランタイムの例
- SDK for PHP を使用した Amazon DocumentDB の例
- SDK for PHP を使用した DynamoDB の例
- SDK for PHP を使用した Amazon EC2 の例
- AWS Glue SDK for PHP を使用した の例
- SDK for PHP を使用した IAM の例
- SDK for PHP を使用した Kinesis の例
- AWS KMS SDK for PHP を使用した の例
- SDK for PHP を使用した Lambda の例
- SDK for PHP を使用した Amazon MSK の例
- SDK for PHP を使用した Amazon RDS の例
- SDK for PHP を使用した Amazon RDS Data Service の例
- SDK for PHP を使用した Amazon Rekognition の例

- SDK for PHP を使用した Amazon S3 のコードサンプル
- SDK for PHP を使用した S3 ディレクトリバケットの例
- SDK for PHP を使用した Amazon SES の例
- SDK for PHP を使用した Amazon SNS のコードサンプル
- SDK for PHP を使用した Amazon SQS の例

SDK for PHP を使用した API Gateway の例

次のコード例は、 API Gateway 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 の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おり、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション
- シナリオ

アクション

`GetBasePathMapping`

次の例は、`GetBasePathMapping` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「GetBasePathMapping」を参照してください。

ListBasePathMappings

次の例は、ListBasePathMappings を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「ListBasePathMappings」を参照してください。

UpdateBasePathMapping

次の例は、UpdateBasePathMapping を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「UpdateBasePathMapping」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。

SDK for PHP

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設定および実行の手順については、GitHub で完全な例を参照してください。

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 Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

SDK for PHP を使用した Aurora の例

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 の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おり、コードの設定方法と実行方法に関する手順を確認できます。

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。

SDK for PHP

を使用して AWS SDK for PHP 、Amazon RDS 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 E メールで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ます。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 RESTful PHP バックエンドと対話します。

- React.js ウェブアプリケーションを AWS サービスに統合します。
- Amazon RDS テーブル内の項目の一覧表示、追加、更新、削除を行います。
- Amazon SES を使用して、フィルター処理された作業項目の E メールレポートを送信します。
- 含まれている 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および実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

SDK for PHP を使用した自動スケーリングの例

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おり、コードの設定方法と実行方法に関する手順を確認できます。

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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「DescribeAutoScalingGroups」を参照してください。

トピック

- 基本
- アクション

基本

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 メトリクスの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 メトリクスの統計を取得して、リソースをクリーンアップします。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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,
        ],
    ]);
}
```

- 詳細については、「AWS SDK for PHP API リファレンス」の「CreateAutoScalingGroup」を参照してください。

`DeleteAutoScalingGroup`

次の例は、`DeleteAutoScalingGroup` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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,
        ]);
    }
```

- 詳細については、「AWS SDK for PHP API リファレンス」の「DeleteAutoScalingGroup」を参照してください。

DescribeAutoScalingGroups

次の例は、DescribeAutoScalingGroups を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「DescribeAutoScalingGroups」を参照してください。

DescribeAutoScalingInstances

次の例は、DescribeAutoScalingInstances を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「DescribeAutoScalingInstances」を参照してください。

`DescribeScalingActivities`

次の例は、`DescribeScalingActivities` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「DescribeScalingActivities」を参照してください。

DisableMetricsCollection

次の例は、DisableMetricsCollection を使用する方法を説明しています。

SDK for PHP

> **Note**
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「DisableMetricsCollection」を参照してください。

EnableMetricsCollection

次の例は、EnableMetricsCollection を使用する方法を説明しています。

SDK for PHP

> **Note**
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「EnableMetricsCollection」を参照してください。

SetDesiredCapacity

次の例は、SetDesiredCapacity を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「SetDesiredCapacity」を参照してください。

TerminateInstanceInAutoScalingGroup

次の例は、TerminateInstanceInAutoScalingGroup を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「TerminateInstanceInAutoScalingGroup」を参照してください。

UpdateAutoScalingGroup

次の例は、UpdateAutoScalingGroup を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「UpdateAutoScalingGroup」を参照してください。

SDK for PHP を使用した Amazon Bedrock の例

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おり、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

アクション

`ListFoundationModels`

次の例は、`ListFoundationModels` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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()
{
    $bedrockClient = new BedrockClient([
        'region' => 'us-west-2',
        'profile' => 'default'
    ]);
    $response = $bedrockClient->listFoundationModels();
    return $response['modelSummaries'];
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「ListFoundationModels」を参照してください。

SDK for PHP を使用した Amazon Bedrock ランタイムの例

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 の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おり、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- シナリオ
- Amazon Nova
- Amazon Titan Image Generator
- Anthropic Claude
- 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) にプロンプトを準備して送信する方法を示しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon Bedrock で複数の LLM 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";
        $bedrockRuntimeService = new BedrockRuntimeService();
        $prompt = 'In one paragraph, who are you?';
        echo "\nPrompt: " . $prompt;
        echo "\n\nAnthropic Claude:\n";
        echo $bedrockRuntimeService->invokeClaude($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 image has 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 image has 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 の詳細については、「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

Amazon Nova

Converse

次のコード例は、Bedrock の Converse API を使用して Amazon Nova にテキストメッセージを送信する方法を示しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Bedrock の Converse API を使用して Amazon Nova にテキストメッセージを送信します。

```
// Use the Conversation API to send a text message to Amazon Nova.

use Aws\BedrockRuntime\BedrockRuntimeClient;
use Aws\Exception\AwsException;
use RuntimeException;
```

```
class Converse
{
    public function converse(): string
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        $client = new BedrockRuntimeClient([
            'region' => 'us-east-1',
            'profile' => 'default'
        ]);

        // Set the model ID, e.g., Amazon Nova Lite.
        $modelId = 'amazon.nova-lite-v1:0';

        // Start a conversation with the user message.
        $userMessage = "Describe the purpose of a 'hello world' program in one
 line.";
        $conversation = [
            [
                "role" => "user",
                "content" => [["text" => $userMessage]]
            ]
        ];

        try {
            // Send the message to the model, using a basic inference configuration.
            $response = $client->converse([
                'modelId' => $modelId,
                'messages' => $conversation,
                'inferenceConfig' => [
                    'maxTokens' => 512,
                    'temperature' => 0.5
                ]
            ]);

            // Extract and return the response text.
            $responseText = $response['output']['message']['content'][0]['text'];
            return $responseText;
        } catch (AwsException $e) {
            echo "ERROR: Can't invoke {$modelId}. Reason: {$e-
>getAwsErrorMessage()}";
            throw new RuntimeException("Failed to invoke model: " . $e-
>getAwsErrorMessage(), 0, $e);
        }
    }
```

```
}

$demo = new Converse();
echo $demo->converse();
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「Converse」を参照してください。

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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon Titan Image Generator を使用して画像を作成します。

```
    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 の詳細については、「AWS SDK for PHP API リファレンス」の「InvokeModel」を参照してください。

Anthropic Claude

InvokeModel

次のコード例は、Invoke Model API 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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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-3-haiku-20240307-v1:0';
        // Claude requires you to enclose the prompt as follows:
            $body = [
                'anthropic_version' => 'bedrock-2023-05-31',
                'max_tokens' => 512,
                'temperature' => 0.5,
                'messages' => [[
                    'role' => 'user',
                    'content' => $prompt
                ]]
            ];
            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $completion = $response_body->content[0]->text;
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }
```

```
        return $completion;
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「InvokeModel」を参照してください。

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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Stable Diffusion で画像を作成します。

```
    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-v1';
            $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 の詳細については、「AWS SDK for PHP API リファレンス」の「InvokeModel」を参照してください。

SDK for PHP を使用した Amazon DocumentDB の例

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おり、コードの設定方法と実行方法に関する手順を確認できます。

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、レコードの内容をログ記録します。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用して Lambda で Amazon DocumentDB イベントの消費。

```
<?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();
```

SDK for PHP を使用した DynamoDB の例

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 の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おり、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本
- アクション
- シナリオ
- サーバーレスサンプル

基本

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 ファイルから映画データをテーブルに書き込む。
- 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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「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`

次の例は、`BatchExecuteStatement` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「BatchExecuteStatement」を参照してください。

BatchWriteItem

次の例は、BatchWriteItem を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「BatchWriteItem」を参照してください。

`CreateTable`

次の例は、`CreateTable` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「CreateTable」を参照してください。

DeleteItem

次の例は、DeleteItem を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteItem」を参照してください。

DeleteTable

次の例は、DeleteTable を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、『AWS SDK for PHP API リファレンス』の「DeleteTable」を参照してください。

`ExecuteStatement`

次の例は、`ExecuteStatement` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ExecuteStatement」を参照してください。

GetItem

次の例は、GetItem を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「GetItem」を参照してください。

ListTables

次の例は、ListTables を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、『AWS SDK for PHP API リファレンス』の「ListTables」を参照してください。

PutItem

次の例は、PutItem を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「PutItem」を参照してください。

Query

次の例は、Query を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「Query」を参照してください。

Scan

次の例は、Scan を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「Scan」を参照してください。

UpdateItem

次の例は、UpdateItem を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「UpdateItem」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。

SDK for PHP

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設定および実行の手順については、GitHub で完全な例を参照してください。

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 Gateway
- 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する。
- 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する。
- 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する。
- 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する。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「BatchExecuteStatement」を参照してください。

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する。

- INSERT 文を実行して項目を追加する。
- UPDATE ステートメントを使用して項目を更新する。
- DELETE ステートメントを実行して項目を削除する。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ExecuteStatement」を参照してください。

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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での DynamoDB イベントの消費。

```
<?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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での DynamoDB バッチ項目失敗のレポート。

```
<?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);
```

SDK for PHP を使用した Amazon EC2 の例

次のコード例は、Amazon EC2 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おり、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

アクション

CreateVpc

次の例は、CreateVpc を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * @param string $cidr
     * @return array
     */
    public function createVpc(string $cidr): array
    {
        try {
            $result = $this->ec2Client->createVpc([
                "CidrBlock" => $cidr,
            ]);
            return $result['Vpc'];
        }catch(Ec2Exception $caught){
            echo "There was a problem creating the VPC: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「CreateVpc」を参照してください。

CreateVpcEndpoint

次の例は、CreateVpcEndpoint を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * @param string $serviceName
     * @param string $vpcId
     * @param array $routeTableIds
     * @return array
     */
    public function createVpcEndpoint(string $serviceName, string $vpcId, array
 $routeTableIds): array
    {
        try {
            $result = $this->ec2Client->createVpcEndpoint([
                'ServiceName' => $serviceName,
                'VpcId' => $vpcId,
                'RouteTableIds' => $routeTableIds,
            ]);

            return $result["VpcEndpoint"];
        } catch(Ec2Exception $caught){
            echo "There was a problem creating the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API の詳細については、AWS SDK for PHP 「 API リファレンス」の「CreateVpcEndpoint」を参照してください。

`DeleteVpc`

次の例は、`DeleteVpc` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * @param string $vpcId
     * @return void
     */
    public function deleteVpc(string $vpcId)
    {
        try {
            $this->ec2Client->deleteVpc([
                "VpcId" => $vpcId,
            ]);
        }catch(Ec2Exception $caught){
            echo "There was a problem deleting the VPC: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API の詳細については、AWS SDK for PHP 「 API リファレンス」のDeleteVpc」を参照してください。

`DeleteVpcEndpoints`

次の例は、`DeleteVpcEndpoints` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * @param string $vpcEndpointId
     * @return void
     */
    public function deleteVpcEndpoint(string $vpcEndpointId)
    {
        try {
            $this->ec2Client->deleteVpcEndpoints([
                "VpcEndpointIds" => [$vpcEndpointId],
            ]);
        }catch (Ec2Exception $caught){
            echo "There was a problem deleting the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API の詳細については、AWS SDK for PHP 「 API リファレンス」のDeleteVpcEndpoints」を参照してください。

`DescribeRouteTables`

次の例は、`DescribeRouteTables` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   /**
    * @param array $routeTableIds
    * @param array $filters
    * @return array
    */
   public function describeRouteTables(array $routeTableIds = [], array $filters =
[]): array
   {
       $parameters = [];
       if($routeTableIds){
           $parameters['RouteTableIds'] = $routeTableIds;
       }
       if($filters){
           $parameters['Filters'] = $filters;
       }
       try {
           $paginator = $this->ec2Client->getPaginator("DescribeRouteTables",
$parameters);
           $contents = [];
           foreach ($paginator as $result) {
               foreach ($result['RouteTables'] as $object) {
                   $contents[] = $object['RouteTableId'];
               }
           }
       }catch (Ec2Exception $caught){
           echo "There was a problem paginating the results of DescribeRouteTables:
{$caught->getAwsErrorMessage()}\n";
           throw $caught;
       }
       return $contents;
   }
```

- API の詳細については、AWS SDK for PHP 「 API リファレンス」の「DescribeRouteTables」を参照してください。

AWS Glue SDK for PHP を使用した の例

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おり、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本
- アクション

基本

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する。
- 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 形式のデータを抽出するジョブを作成し、そのデータを変換して JSON 形式の出力を別の S3 バケットにロードする。
- 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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「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

次の例は、CreateCrawler を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「CreateCrawler」を参照してください。

`CreateJob`

次の例は、`CreateJob` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「CreateJob」を参照してください。

DeleteCrawler

次の例は、DeleteCrawler を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteCrawler」を参照してください。

DeleteDatabase

次の例は、DeleteDatabase を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteDatabase」を参照してください。

`DeleteJob`

次の例は、`DeleteJob` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteJob」を参照してください。

DeleteTable

次の例は、DeleteTable を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、『AWS SDK for PHP API リファレンス』の「DeleteTable」を参照してください。

GetCrawler

次の例は、GetCrawler を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「GetCrawler」を参照してください。

GetDatabase

次の例は、GetDatabase を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「GetDatabase」を参照してください。

`GetJobRun`

次の例は、`GetJobRun` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「GetJobRun」を参照してください。

GetJobRuns

次の例は、GetJobRuns を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「GetJobRuns」を参照してください。

GetTables

次の例は、GetTables を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「GetTables」を参照してください。

ListJobs

次の例は、ListJobs を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListJobs」を参照してください。

StartCrawler

次の例は、StartCrawler を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「StartCrawler」を参照してください。

StartJobRun

次の例は、StartJobRun を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「StartJobRun」を参照してください。

SDK for PHP を使用した IAM の例

次のコード例は、IAM 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おり、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本
- アクション

基本

基本を学ぶ

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**
> セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に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。

SDK for PHP

> **Note**
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「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

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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「AttachRolePolicy」を参照してください。

CreatePolicy

次の例は、CreatePolicy を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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";

    /**
     * @param string $policyName
     * @param string $policyDocument
     * @return array
     */
    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 の詳細については、「AWS SDK for PHP API リファレンス」の「CreatePolicy」を参照してください。

CreateRole

次の例は、CreateRole を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「CreateRole」を参照してください。

`CreateServiceLinkedRole`

次の例は、`CreateServiceLinkedRole` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「CreateServiceLinkedRole」を参照してください。

`CreateUser`

次の例は、`CreateUser` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「CreateUser」を参照してください。

GetAccountPasswordPolicy

次の例は、GetAccountPasswordPolicy を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「GetAccountPasswordPolicy」を参照してください。

GetPolicy

次の例は、GetPolicy を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「GetPolicy」を参照してください。

`GetRole`

次の例は、`GetRole` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「GetRole」を参照してください。

ListAttachedRolePolicies

次の例は、ListAttachedRolePolicies を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListAttachedRolePolicies」を参照してください。

ListGroups

次の例は、ListGroups を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListGroup」を参照してください。

ListPolicies

次の例は、ListPolicies を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListPolicies」を参照してください。

ListRolePolicies

次の例は、ListRolePolicies を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListRolePolicies」を参照してください。

`ListRoles`

次の例は、`ListRoles` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListRoles」を参照してください。

ListSAMLProviders

次の例は、ListSAMLProviders を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListSAMLProviders」を参照してください。

ListUsers

次の例は、ListUsers を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListUsers」を参照してください。

SDK for PHP を使用した Kinesis の例

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おり、コードの設定方法と実行方法に関する手順を確認できます。

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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での Kinesis イベントの消費。

```
// 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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での Kinesis バッチアイテム失敗のレポート。

```
// 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);
```

AWS KMS SDK for PHP を使用した の例

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 KMS。

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おり、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

こんにち AWS Key Management Serviceは

次のコード例は、 AWS Key Management Serviceの使用を開始する方法を示しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
include "vendor/autoload.php";

use Aws\Kms\KmsClient;

echo "This file shows how to connect to the KmsClient, uses a paginator to get the
 keys for the account, and lists the KeyIds for up to 10 keys.\n";

$client = new KmsClient([]);

$pageLength = 10; // Change this value to change the number of records shown, or to
 break up the result into pages.

$keys = [];
$keysPaginator = $client->getPaginator("ListKeys", ['Limit' => $pageLength]);
foreach($keysPaginator as $page){
    foreach($page['Keys'] as $index => $key){
        echo "The $index index Key's ID is: {$key['KeyId']}\n";
    }
    echo "End of page one of results. Alter the \$pageLength variable to see more
 results.\n";
    break;
```

```
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「ListKeys」を参照してください。

トピック

- 基本
- アクション

基本

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。

- KMS キーを作成します。
- アカウントの KMS キーを一覧表示し、詳細を取得します。
- KMS キーを有効および無効にします。
- クライアント側の暗号化に使用できる対称データキーを生成します。
- データのデジタル署名に使用する非対称キーを生成します。
- キーをタグ付けします。
- KMS キーを削除します。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    echo "\n";
    echo "--------------------------------------\n";
```

```
        echo <<<WELCOME
Welcome to the AWS Key Management Service SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management Service using the
 AWS SDK for PHP (v3).
The AWS Key Management Service (KMS) is a secure and highly available service that
 allows you to create
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys, making
 it easier to meet your
data protection and regulatory compliance requirements.

This KMS Basics scenario creates two key types:
- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...\n
WELCOME;
        echo "--------------------------------------\n";
        $this->pressEnter();

        $this->kmsClient = new KmsClient([]);
        // Initialize the KmsService class with the client. This allows you to
 override any defaults in the client before giving it to the service class.
        $this->kmsService = new KmsService($this->kmsClient);

        // 1. Create a symmetric KMS key.
        echo "\n";
        echo "1. Create a symmetric KMS key.\n";
        echo "First, we will create a symmetric KMS key that is used to encrypt and
 decrypt data by invoking createKey().\n";
        $this->pressEnter();

        $key = $this->kmsService->createKey();
        $this->resources['symmetricKey'] = $key['KeyId'];
        echo "Created a customer key with ARN {$key['Arn']}.\n";
        $this->pressEnter();

        // 2. Enable a KMS key.
        echo "\n";
        echo "2. Enable a KMS key.\n";
        echo "By default when you create an AWS key, it is enabled. The code checks
 to
```

```
determine if the key is enabled. If it is not enabled, the code enables it.\n";
        $this->pressEnter();

        $keyInfo = $this->kmsService->describeKey($key['KeyId']);
        if(!$keyInfo['Enabled']){
            echo "The key was not enabled, so we will enable it.\n";
            $this->pressEnter();
            $this->kmsService->enableKey($key['KeyId']);
            echo "The key was successfully enabled.\n";
        }else{
            echo "The key was already enabled, so there was no need to enable it.
\n";
        }
        $this->pressEnter();

        // 3. Encrypt data using the symmetric KMS key.
        echo "\n";
        echo "3. Encrypt data using the symmetric KMS key.\n";
        echo "One of the main uses of symmetric keys is to encrypt and decrypt data.
\n";
        echo "Next, we'll encrypt the string 'Hello, AWS KMS!' with the
 SYMMETRIC_DEFAULT encryption algorithm.\n";
        $this->pressEnter();
        $text = "Hello, AWS KMS!";
        $encryption = $this->kmsService->encrypt($key['KeyId'], $text);
        echo "The plaintext data was successfully encrypted with the algorithm:
 {$encryption['EncryptionAlgorithm']}.\n";
        $this->pressEnter();

        // 4. Create an alias.
        echo "\n";
        echo "4. Create an alias.\n";
        $aliasInput = testable_readline("Please enter an alias prefixed with
 \"alias/\" or press enter to use a default value: ");
        if($aliasInput == ""){
            $aliasInput = "alias/dev-encryption-key";
        }
        $this->kmsService->createAlias($key['KeyId'], $aliasInput);
        $this->resources['alias'] = $aliasInput;
        echo "The alias \"$aliasInput\" was successfully created.\n";
        $this->pressEnter();

        // 5. List all of your aliases.
        $aliasPageSize = 10;
```

```
        echo "\n";
        echo "5. List all of your aliases, up to $aliasPageSize.\n";
        $this->pressEnter();
        $aliasPaginator = $this->kmsService->listAliases();
        foreach($aliasPaginator as $pages){
            foreach($pages['Aliases'] as $alias){
                echo $alias['AliasName'] . "\n";
            }
            break;
        }
        $this->pressEnter();

        // 6. Enable automatic rotation of the KMS key.
        echo "\n";
        echo "6. Enable automatic rotation of the KMS key.\n";
        echo "By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days) from
 the enable date and every year
thereafter.";
        $this->pressEnter();
        $this->kmsService->enableKeyRotation($key['KeyId']);
        echo "The key's rotation was successfully set for key: {$key['KeyId']}\n";
        $this->pressEnter();

        // 7. Create a grant.
        echo "7. Create a grant.\n";
        echo "\n";
        echo "A grant is a policy instrument that allows Amazon Web Services
 principals to use KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage grants.
When authorizing access to a KMS key, grants are considered along with key policies
 and IAM policies.\n";
        $granteeARN = testable_readline("Please enter the Amazon Resource Name
 (ARN) of an Amazon Web Services principal. Valid principals include Amazon Web
 Services accounts, IAM users, IAM roles, federated users, and assumed role users.
 For help with the ARN syntax for a principal, see IAM ARNs in the Identity and
 Access Management User Guide. \nTo skip this step, press enter without any other
 values: ");
        if($granteeARN){
            $operations = [
                "ENCRYPT",
                "DECRYPT",
                "DESCRIBE_KEY",
            ];
```

```
            $grant = $this->kmsService->createGrant($key['KeyId'], $granteeARN,
$operations);
            echo "The grant Id is: {$grant['GrantId']}\n";
        }else{
            echo "Steps 7, 8, and 9 will be skipped.\n";
        }
        $this->pressEnter();

        // 8. List grants for the KMS key.
        if($granteeARN){
            echo "8. List grants for the KMS key.\n\n";
            $grantsPaginator = $this->kmsService->listGrants($key['KeyId']);
            foreach($grantsPaginator as $page){
                foreach($page['Grants'] as $grant){
                    echo $grant['GrantId'] . "\n";
                }
            }
        }else{
            echo "Skipping step 8...\n";
        }
        $this->pressEnter();

        // 9. Revoke the grant.
        if($granteeARN) {
            echo "\n";
            echo "9. Revoke the grant.\n";
            $this->pressEnter();
            $this->kmsService->revokeGrant($grant['GrantId'], $keyInfo['KeyId']);
            echo "{$grant['GrantId']} was successfully revoked!\n";
        }else{
            echo "Skipping step 9...\n";
        }
        $this->pressEnter();

        // 10. Decrypt the data.
        echo "\n";
        echo "10. Decrypt the data.\n";
        echo "Let's decrypt the data that was encrypted before.\n";
        echo "We'll use the same key to decrypt the string that we encrypted earlier
in the program.\n";
        $this->pressEnter();
        $decryption = $this->kmsService->decrypt($keyInfo['KeyId'],
$encryption['CiphertextBlob'], $encryption['EncryptionAlgorithm']);
        echo "The decrypted text is: {$decryption['Plaintext']}\n";
```

```
        $this->pressEnter();

        // 11. Replace a Key Policy.
        echo "\n";
        echo "11. Replace a Key Policy.\n";
        echo "A key policy is a resource policy for a KMS key. Key policies are the
 primary way to control access to KMS keys.\n";
        echo "Every KMS key must have exactly one key policy. The statements in the
 key policy determine who has permission to use the KMS key and how they can use it.
\n";
        echo " You can also use IAM policies and grants to control access to the KMS
 key, but every KMS key must have a key policy.\n";
        echo "We will replace the key's policy with a new one:\n";
        $stsClient = new StsClient([]);
        $result = $stsClient->getCallerIdentity();
        $accountId = $result['Account'];
        $keyPolicy = <<< KEYPOLICY
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::$accountId:root"},
        "Action": "kms:*",
        "Resource": "*"
    }]
}
KEYPOLICY;
        echo $keyPolicy;
        $this->pressEnter();
        $this->kmsService->putKeyPolicy($keyInfo['KeyId'], $keyPolicy);
        echo "The Key Policy was successfully replaced!\n";
        $this->pressEnter();

        // 12. Retrieve the key policy.
        echo "\n";
        echo "12. Retrieve the key policy.\n";
        echo "Let's get some information about the new policy and print it to the
 screen.\n";
        $this->pressEnter();
        $policyInfo = $this->kmsService->getKeyPolicy($keyInfo['KeyId']);
        echo "We got the info! Here is the policy: \n";
        echo $policyInfo['Policy'] . "\n";
        $this->pressEnter();
```

```
        // 13. Create an asymmetric KMS key and sign data.
        echo "\n";
        echo "13. Create an asymmetric KMS key and sign data.\n";
        echo "Signing your data with an AWS key can provide several benefits that
make it an attractive option for your data signing needs.\n";
        echo "By using an AWS KMS key, you can leverage the security controls and
compliance features provided by AWS, which can help you meet various regulatory
requirements and enhance the overall security posture of your organization.\n";
        echo "First we'll create the asymmetric key.\n";
        $this->pressEnter();
        $keySpec = "RSA_2048";
        $keyUsage = "SIGN_VERIFY";
        $asymmetricKey = $this->kmsService->createKey($keySpec, $keyUsage);
        $this->resources['asymmetricKey'] = $asymmetricKey['KeyId'];
        echo "Created the key with ID: {$asymmetricKey['KeyId']}\n";
        echo "Next, we'll sign the data.\n";
        $this->pressEnter();
        $algorithm = "RSASSA_PSS_SHA_256";
        $sign = $this->kmsService->sign($asymmetricKey['KeyId'], $text, $algorithm);
        $verify = $this->kmsService->verify($asymmetricKey['KeyId'], $text,
$sign['Signature'], $algorithm);
        echo "Signature verification result: {$sign['signature']}\n";
        $this->pressEnter();

        // 14. Tag the symmetric KMS key.
        echo "\n";
        echo "14. Tag the symmetric KMS key.\n";
        echo "By using tags, you can improve the overall management, security,
and governance of your KMS keys, making it easier to organize, track, and control
access to your encrypted data within your AWS environment.\n";
        echo "Let's tag our symmetric key as Environment->Production\n";
        $this->pressEnter();
        $this->kmsService->tagResource($key['KeyId'], [
            [
                'TagKey' => "Environment",
                'TagValue' => "Production",
            ],
        ]);
        echo "The key was successfully tagged!\n";
        $this->pressEnter();

        // 15. Schedule the deletion of the KMS key
        echo "\n";
        echo "15. Schedule the deletion of the KMS key.\n";
```

```
        echo "By default, KMS applies a waiting period of 30 days, but you can
 specify a waiting period of 7-30 days.\n";
        echo "When this operation is successful, the key state of the KMS key
 changes to PendingDeletion and the key can't be used in any cryptographic
 operations.\n";
        echo "It remains in this state for the duration of the waiting period.\n\n";

        echo "Deleting a KMS key is a destructive and potentially dangerous
 operation. When a KMS key is deleted, all data that was encrypted under the KMS key
 is unrecoverable.\n\n";

        $cleanUp = testable_readline("Would you like to delete the resources created
 during this scenario, including the keys? (y/n): ");
        if($cleanUp == "Y" || $cleanUp == "y"){
            $this->cleanUp();
        }

        echo
 "--------------------------------------------------------------------------------
\n";
        echo "This concludes the AWS Key Management SDK Basics scenario\n";
        echo
 "--------------------------------------------------------------------------------
\n";



namespace Kms;

use Aws\Kms\Exception\KmsException;
use Aws\Kms\KmsClient;
use Aws\Result;
use Aws\ResultPaginator;
use AwsUtilities\AWSServiceClass;

class KmsService extends AWSServiceClass
{

    protected KmsClient $client;
    protected bool $verbose;

    /***
     * @param KmsClient|null $client
     * @param bool $verbose
```

```
     */
    public function __construct(KmsClient $client = null, bool $verbose = false)
    {
        $this->verbose = $verbose;
        if($client){
            $this->client = $client;
            return;
        }
        $this->client = new KmsClient([]);
    }


    /***
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
 $description = "Created by the SDK for PHP")
    {
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
 more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $ciphertext
```

```
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm =
 "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is not
 enabled.\n";
            }
            throw $caught;
        }
    }
```

```
    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name is
not valid.";
            }
```

```
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
    public function createGrant(string $keyId, string $granteePrincipal, array
 $operations, array $grantTokens = [])
    {
        $args = [
            'KeyId' => $keyId,
            'GranteePrincipal' => $granteePrincipal,
            'Operations' => $operations,
        ];
        if($grantTokens){
            $args['GrantTokens'] = $grantTokens;
        }
        try{
            return $this->client->createGrant($args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
                echo "The request was rejected because the specified grant token is
 not valid.\n";
            }
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
```

```
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
```

```
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
```

```
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }


    /***
     * @param string $keyId
     * @return Result
     */
    public function getKeyPolicy(string $keyId)
    {
        try {
            return $this->client->getKeyPolicy([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem getting the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
```

```
        }
    }



    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays = 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

```
    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $rotationPeriodInDays
     * @return void
     */
    public function enableKeyRotation(string $keyId, int $rotationPeriodInDays =
 365)
    {
        try{
            $this->client->enableKeyRotation([
                'KeyId' => $keyId,
                'RotationPeriodInDays' => $rotationPeriodInDays,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
```

```
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }




    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

```
    /***
     * @param string $keyId
     * @param string $message
     * @param string $signature
     * @param string $signingAlgorithm
     * @return bool
     */
    public function verify(string $keyId, string $message, string $signature, string
 $signingAlgorithm)
    {
        try {
            $result = $this->client->verify([
                'KeyId' => $keyId,
                'Message' => $message,
                'Signature' => $signature,
                'SigningAlgorithm' => $signingAlgorithm,
            ]);
            return $result['SignatureValid'];
        }catch(KmsException $caught){
            echo "There was a problem verifying the signature: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


}
```

- API の詳細については、『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。
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey

- EnableKey
- 暗号化
- GetKeyPolicy
- ListAliases
- ListGrants
- ListKeys
- RevokeGrant
- ScheduleKeyDeletion
- Sign
- TagResource

アクション

`CreateAlias`

次の例は、`CreateAlias` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/***
 * @param string $keyId
 * @param string $alias
 * @return void
 */
public function createAlias(string $keyId, string $alias)
{
    try{
        $this->client->createAlias([
            'TargetKeyId' => $keyId,
            'AliasName' => $alias,
```

```
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name is
not valid.";
            }
            throw $caught;
        }
    }
```

- API の詳細については、AWS SDK for PHP API リファレンスのCreateAliasを参照してください。

CreateGrant

次の例は、CreateGrant を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keyId
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
    public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
    {
        $args = [
            'KeyId' => $keyId,
            'GranteePrincipal' => $granteePrincipal,
```

```
            'Operations' => $operations,
        ];
        if($grantTokens){
            $args['GrantTokens'] = $grantTokens;
        }
        try{
            return $this->client->createGrant($args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
                echo "The request was rejected because the specified grant token is
not valid.\n";
            }
            throw $caught;
        }
    }
```

- API の詳細については、AWS SDK for PHP API リファレンスのCreateGrantを参照してください。

CreateKey

次の例は、CreateKey を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
```

```
    public function createKey(string $keySpec = "", string $keyUsage = "", string
$description = "Created by the SDK for PHP")
    {
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「CreateKey」を参照してください。

Decrypt

次の例は、Decrypt を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keyId
```

```
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm =
 "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「Decrypt」を参照してください。

DeleteAlias

次の例は、DeleteAlias を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $aliasName
     * @return void
```

```
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteAlias」を参照してください。

DescribeKey

次の例は、DescribeKey を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
```

```
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- 詳細については、AWS SDK for PHP API リファレンスの DescribeKeyを参照してください。

DisableKey

次の例は、DisableKey を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keyId
     * @return void
     */
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
```

```
            throw $caught;
        }
    }
```

- 詳細については、AWS SDK for PHP API リファレンスの DisableKeyを参照してください。

EnableKey

次の例は、EnableKey を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「EnableKey」を参照してください。

Encrypt

次の例は、Encrypt を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is not
enabled.\n";
            }
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「Encrypt」を参照してください。

ListAliases

次の例は、ListAliases を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「ListAliases」を参照してください。

ListGrants

次の例は、ListGrants を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「ListGrants」を参照してください。

ListKeys

次の例は、ListKeys を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「ListKeys」を参照してください。

PutKeyPolicy

次の例は、PutKeyPolicy を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「PutKeyPolicy」を参照してください。

RevokeGrant

次の例は、RevokeGrant を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「RevokeGrant」を参照してください。

`ScheduleKeyDeletion`

次の例は、`ScheduleKeyDeletion` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays = 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「ScheduleKeyDeletion」を参照してください。

Sign

次の例は、Sign を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「署名」を参照してください。

`TagResource`

次の例は、`TagResource` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /***
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「TagResource」を参照してください。

SDK for PHP を使用した Lambda の例

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。

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 の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おり、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本
- アクション
- シナリオ
- サーバーレスサンプル

基本

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。

- 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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、『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

アクション

`CreateFunction`

次の例は、`CreateFunction` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「CreateFunction」を参照してください。

DeleteFunction

次の例は、DeleteFunction を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteFunction」を参照してください。

GetFunction

次の例は、GetFunction を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「GetFunction」を参照してください。

Invoke

次の例は、Invoke を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「Invoke」を参照してください。

ListFunctions

次の例は、ListFunctions を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListFunctions」を参照してください。

UpdateFunctionCode

次の例は、UpdateFunctionCode を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「UpdateFunctionCode」を参照してください。

UpdateFunctionConfiguration

次の例は、UpdateFunctionConfiguration を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「UpdateFunctionConfiguration」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。

SDK for PHP

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設定および実行の手順については、GitHub で完全な例を参照してください。

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 Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

サーバーレスサンプル

Lambda 関数での 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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda 関数での Amazon RDS データベースへの接続

```
<?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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での Kinesis イベントの消費。

```
// 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、レコードの内容をログ記録します。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での DynamoDB イベントの消費。

```
<?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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用して Lambda で Amazon DocumentDB イベントの消費。

```
<?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 MSK トリガーから Lambda 関数を呼び出す

次のコード例は、Amazon MSK クラスター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 MSK ペイロードを取得し、レコードの内容をログ記録します。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での Amazon MSK イベントの消費。

```
<?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\Kafka\KafkaEvent;
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): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

        foreach ($records as $record) {
            try {
                $key = $record->getKey();
                $this->logger->info("Key: $key");

                $values = $record->getValue();
                $this->logger->info(json_encode($values));

                foreach ($values as $value) {
                    $this->logger->info("Value: $value");
                }

            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

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 API を呼び出し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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用して Lambda で S3 イベントの消費。

```
<?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メッセージを取得し、各メッセージの内容を記録します。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用して Lambda で SNS イベントを消費します。

```
// 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

次のコード例では、SQS 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け取る、Lambda 関数の実装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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での SQS イベントの消費。

```
// 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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での Kinesis バッチアイテム失敗のレポート。

```
// 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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での DynamoDB バッチ項目失敗のレポート。

```
<?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、SQS キュー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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での SQS バッチアイテム失敗のレポート。

```
// 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 for PHP を使用した Amazon MSK の例

次のコード例は、Amazon MSK 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おり、コードの設定方法と実行方法に関する手順を確認できます。

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 MSK トリガーから Lambda 関数を呼び出す

次のコード例は、Amazon MSK クラスター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 MSK ペイロードを取得し、レコードの内容をログ記録します。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での Amazon MSK イベントの消費。

```
<?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\Kafka\KafkaEvent;
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): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

        foreach ($records as $record) {
            try {
```

```
                $key = $record->getKey();
                $this->logger->info("Key: $key");

                $values = $record->getValue();
                $this->logger->info(json_encode($values));

                foreach ($values as $value) {
                    $this->logger->info("Value: $value");
                }

            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

SDK for PHP を使用した Amazon RDS の例

次のコード例は、Amazon RDS 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 の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おり、コードの設定方法と実行方法に関する手順を確認できます。

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

`CreateDBInstance`

次の例は、`CreateDBInstance` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「CreateDBInstance」を参照してください。

CreateDBSnapshot

次の例は、CreateDBSnapshot を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「CreateDBSnapshot」を参照してください。

`DeleteDBInstance`

次の例は、`DeleteDBInstance` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteDBInstance」を参照してください。

DescribeDBInstances

次の例は、DescribeDBInstances を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「DescribeDBInstances」を参照してください。

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。

SDK for PHP

を使用して AWS SDK for PHP 、Amazon RDS 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 E メールで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ます。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 RESTful PHP バックエンドと対話します。

- React.js ウェブアプリケーションを AWS サービスに統合します。
- Amazon RDS テーブル内の項目の一覧表示、追加、更新、削除を行います。
- Amazon SES を使用して、フィルター処理された作業項目の E メールレポートを送信します。
- 含まれている 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および実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

サーバーレスサンプル

Lambda 関数での 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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda 関数での Amazon RDS データベースへの接続

```
<?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 for PHP を使用した Amazon RDS Data Service の例

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 の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おり、コードの設定方法と実行方法に関する手順を確認できます。

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。

SDK for PHP

を使用して AWS SDK for PHP 、Amazon RDS 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 E メールで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ます。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 RESTful PHP バックエンドと対話します。

- React.js ウェブアプリケーションを AWS サービスに統合します。
- Amazon RDS テーブル内の項目の一覧表示、追加、更新、削除を行います。
- Amazon SES を使用して、フィルター処理された作業項目の E メールレポートを送信します。
- 含まれている 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および実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

SDK for PHP を使用した Amazon Rekognition の例

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。

「シナリオ」は、1 つのサービス内から、または他の AWS の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おり、コードの設定方法と実行方法に関する手順を確認できます。

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。

SDK for PHP

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設定および実行の手順については、GitHub で完全な例を参照してください。

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 Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

SDK for PHP を使用した Amazon S3 のコードサンプル

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 の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おり、コードの設定方法と実行方法に関する手順を確認できます。

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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListBuckets」を参照してください。

トピック

- 基本
- アクション
- シナリオ
- サーバーレスサンプル

基本

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。

SDK for PHP

> **Note**
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 = "amzn-s3-demo-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 (isset($check['Contents']) && count($check['Contents']) > 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 の詳細については、「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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「CopyObject」を参照してください。

CreateBucket

次の例は、CreateBucket を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「CreateBucket」を参照してください。

`DeleteBucket`

次の例は、`DeleteBucket` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteBucket」を参照してください。

DeleteObject

次の例は、DeleteObject を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   public function deleteObject(string $bucketName, string $fileName, array $args =
[])
   {
       $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
       try {
           $this->client->deleteObject($parameters);
           if ($this->verbose) {
               echo "Deleted the object named: $fileName from $bucketName.\n";
           }
       } catch (AwsException $exception) {
```

```
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteObject」を参照してください。

DeleteObjects

次の例は、DeleteObjects を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 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 の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteObjects」を参照してください。

GetObject

次の例は、GetObject を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「GetObject」を参照してください。

ListObjectsV2

次の例は、ListObjectsV2 を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListObjectsV2」を参照してください。

PutObject

次の例は、PutObject を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「PutObject」を参照してください。

シナリオ

署名付き URL を作成する

次のコード例は、Amazon S3 の署名付き URL 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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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();



namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
```

```
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }

    public function setClient(S3Client $client)
    {
        $this->client = $client;
    }


    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
```

```
                echo "\nPlease fix error with bucket deletion before continuing.\n";
            }
            throw $exception;
        }
    }



    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
        }
    }



    public function putObject(string $bucketName, string $key, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $this->client->putObject($parameters);
            if ($this->verbose) {
                echo "Uploaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $key in $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with object uploading before continuing.";
            }
            throw $exception;
```

```
        }
    }



    public function getObject(string $bucketName, string $key, array $args = []):
 Result
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $object = $this->client->getObject($parameters);
            if ($this->verbose) {
                echo "Downloaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to download $key from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object downloading before continuing.";
            }
            throw $exception;
        }
        return $object;
    }



    public function copyObject($bucketName, $key, $copySource, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
 "CopySource" => $copySource], $args);
        try {
            $this->client->copyObject($parameters);
            if ($this->verbose) {
                echo "Copied the object from: $copySource in $bucketName to: $key.
\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to copy $copySource in $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object copying before continuing.";
            }
```

```
            throw $exception;
        }
    }



    public function listObjects(string $bucketName, $start = 0, $max = 1000, array
$args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
            throw $exception;
        }
        return $objects;
    }



    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
```

```
        return $contents;
    }



    public function deleteObjects(string $bucketName, array $objects, array $args =
[])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' => ['Objects'
=> $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
```

```
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteObject(string $bucketName, string $fileName, array $args =
 [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
```

```
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }



    public function preSignedUrl(CommandInterface $command, DateTimeInterface|int|
string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
```

```
    }

}
```

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。

SDK for PHP

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設定および実行の手順については、GitHub で完全な例を参照してください。

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 Gateway
- 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 API を呼び出し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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用して Lambda で S3 イベントの消費。

```
<?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 for PHP を使用した S3 ディレクトリバケットの例

次のコード例は、S3 ディレクトリバケット 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。

各例には、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本

基本

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。

- VPC と VPC エンドポイントを設定します。
- S3 ディレクトリバケットと S3 S3 Express One Zone ストレージクラスを操作するようにポリシー、ロール、ユーザーを設定します。
- 2 つの S3 クライアントを作成します。

- バケットを 2 つ作成します。
- オブジェクトを作成してコピーします。
- パフォーマンスの違いを示します。
- 辞書の違いを表示するには、バケットを入力します。
- リソースをクリーンアップするかどうかをユーザーに確認します。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon S3 ディレクトリバケットと S3 Express One Zone の基本を示すシナリオを実行します。

```
        echo "\n";
        echo "-------------------------------------\n";
        echo "Welcome to the Amazon S3 Express Basics demo using PHP!\n";
        echo "-------------------------------------\n";

        // Change these both of these values to use a different region/availability
 zone.
        $region = "us-west-2";
        $az = "usw2-az1";

        $this->s3Service = new S3Service(new S3Client(['region' => $region]));
        $this->iamService = new IAMService(new IamClient(['region' => $region]));

        $uuid = uniqid();

        echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
 working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results in
 this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute resources
 in the same AZ as your Directory
bucket.\n
```

```
INTRO;
        pressEnter();
        // 1. Configure a gateway VPC endpoint. This is the recommended method to
 allow S3 Express One Zone traffic without
        // the need to pass through an internet gateway or NAT device.
        echo "\n";
        echo "1. First, we'll set up a new VPC and VPC Endpoint if this program is
 running in an EC2 instance in the same AZ as your Directory buckets will be.\n";
        $ec2Choice = testable_readline("Are you running this in an EC2 instance
 located in the same AZ as your intended Directory buckets? Enter Y/y to setup a VPC
 Endpoint, or N/n/blank to skip this section.");
        if($ec2Choice == "Y" || $ec2Choice == "y") {
            echo "Great! Let's set up a VPC, retrieve the Route Table from it, and
 create a VPC Endpoint to connect the S3 Client to.\n";
            pressEnter();
            $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
 $region]));
            $cidr = "10.0.0.0/16";
            $vpc = $this->ec2Service->createVpc($cidr);
            $this->resources['vpcId'] = $vpc['VpcId'];

            $this->ec2Service->waitForVpcAvailable($vpc['VpcId']);

            $routeTable = $this->ec2Service->describeRouteTables([], [
                [
                    'Name' => "vpc-id",
                    'Values' => [$vpc['VpcId']],
                ],
            ]);

            $serviceName = "com.amazonaws." . $this->ec2Service->getRegion() .
 ".s3express";
            $vpcEndpoint = $this->ec2Service->createVpcEndpoint($serviceName,
 $vpc['VpcId'], [$routeTable[0]]);
            $this->resources['vpcEndpointId'] = $vpcEndpoint['VpcEndpointId'];
        }else{
            echo "Skipping the VPC setup. Don't forget to use this in production!
\n";
        }

        // 2. Policies, user, and roles with CDK.
        echo "\n";
        echo "2. Policies, users, and roles with CDK.\n";
```

```
        echo "Now, we'll set up some policies, roles, and a user. This user will
 only have permissions to do S3 Express One Zone actions.\n";
        pressEnter();

        $this->cloudFormationClient = new CloudFormationClient([]);
        $stackName = "cfn-stack-s3-express-basics-" . uniqid();
        $file = file_get_contents(__DIR__ . "/../../../../resources/cfn/
s3_express_basics/s3_express_template.yml");
        $result = $this->cloudFormationClient->createStack([
            'StackName' => $stackName,
            'TemplateBody' => $file,
            'Capabilities' => ['CAPABILITY_IAM'],
        ]);
        $waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
 ['StackName' => $stackName]);
        try {
            $waiter->promise()->wait();
        }catch(CloudFormationException $caught){
            echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
        $this->resources['stackName'] = $stackName;
        $stackInfo = $this->cloudFormationClient->describeStacks([
            'StackName' => $result['StackId'],
        ]);

        $expressUserName = "";
        $regularUserName = "";
        foreach($stackInfo['Stacks'][0]['Outputs'] as $output) {
            if ($output['OutputKey'] == "RegularUser") {
                $regularUserName = $output['OutputValue'];
            }
            if ($output['OutputKey'] == "ExpressUser") {
                $expressUserName = $output['OutputValue'];
            }
        }
        $regularKey = $this->iamService->createAccessKey($regularUserName);
        $regularCredentials = new Credentials($regularKey['AccessKeyId'],
 $regularKey['SecretAccessKey']);
        $expressKey = $this->iamService->createAccessKey($expressUserName);
        $expressCredentials = new Credentials($expressKey['AccessKeyId'],
 $expressKey['SecretAccessKey']);
```

```
        // 3. Create an additional client using the credentials with S3 Express
permissions.
        echo "\n";
        echo "3. Create an additional client using the credentials with S3 Express
permissions.\n";
        echo "This client is created with the credentials associated with the
user account with the S3 Express policy attached, so it can perform S3 Express
operations.\n";
        pressEnter();
        $s3RegularClient = new S3Client([
            'Region' => $region,
            'Credentials' => $regularCredentials,
        ]);
        $s3RegularService = new S3Service($s3RegularClient);
        $s3ExpressClient = new S3Client([
            'Region' => $region,
            'Credentials' => $expressCredentials,
        ]);
        $s3ExpressService = new S3Service($s3ExpressClient);
        echo "All the roles and policies were created an attached to the user. Then,
a new S3 Client and Service were created using that user's credentials.\n";
        echo "We can now use this client to make calls to S3 Express operations.
Keeping permissions in mind (and adhering to least-privilege) is crucial to S3
Express.\n";
        pressEnter();

        // 4. Create two buckets.
        echo "\n";
        echo "3. Create two buckets.\n";
        echo "Now we will create a Directory bucket, which is the linchpin of the S3
Express One Zone service.\n";
        echo "Directory buckets behave in different ways from regular S3 buckets,
which we will explore here.\n";
        echo "We'll also create a normal bucket, put an object into the normal
bucket, and copy it over to the Directory bucket.\n";
        pressEnter();

        // Create a directory bucket. These are different from normal S3 buckets in
subtle ways.
        $directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-s3";
        echo "Now, let's create the actual Directory bucket, as well as a regular
bucket.\n";
        pressEnter();
        $s3ExpressService->createBucket($directoryBucketName, [
```

```
            'CreateBucketConfiguration' => [
                'Bucket' => [
                    'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
                    'DataRedundancy' => "SingleAvailabilityZone",
                ],
                'Location' => [
                    'Name' => $az,
                    'Type' => "AvailabilityZone",
                ],
            ],
        ]);
        $this->resources['directoryBucketName'] = $directoryBucketName;

        // Create a normal bucket.
        $normalBucketName = "normal-bucket-$uuid";
        $s3RegularService->createBucket($normalBucketName);
        $this->resources['normalBucketName'] = $normalBucketName;
        echo "Great! Both buckets were created.\n";
        pressEnter();

        // 5. Create an object and copy it over.
        echo "\n";
        echo "5. Create an object and copy it over.\n";
        echo "We'll create a basic object consisting of some text and upload it to
the normal bucket.\n";
        echo "Next, we'll copy the object into the Directory bucket using the
regular client.\n";
        echo "This works fine, because Copy operations are not restricted for
Directory buckets.\n";
        pressEnter();

        $objectKey = "basic-text-object";
        $s3RegularService->putObject($normalBucketName, $objectKey, $args = ['Body'
=> "Look Ma, I'm a bucket!"]);
        $this->resources['objectKey'] = $objectKey;

        // Create a session to access the directory bucket. The SDK Client will
automatically refresh this as needed.
        $s3ExpressService->createSession($directoryBucketName);
        $s3ExpressService->copyObject($directoryBucketName, $objectKey,
"$normalBucketName/$objectKey");
```

```
        echo "It worked! It's important to remember the user permissions when
interacting with Directory buckets.\n";
        echo "Instead of validating permissions on every call as normal buckets do,
Directory buckets utilize the user credentials and session token to validate.\n";
        echo "This allows for much faster connection speeds on every call. For
single calls, this is low, but for many concurrent calls, this adds up to a lot of
time saved.\n";
        pressEnter();

        // 6. Demonstrate performance difference.
        echo "\n";
        echo "6. Demonstrate performance difference.\n";
        $downloads = 1000;
        echo "Now, let's do a performance test. We'll download the same object
from each bucket $downloads times and compare the total time needed. Note: the
performance difference will be much more pronounced if this example is run in an
EC2 instance in the same AZ as the bucket.\n";
        $downloadChoice = testable_readline("If you would like to download each
object $downloads times, press enter. Otherwise, enter a custom amount and press
enter.");
        if($downloadChoice && is_numeric($downloadChoice) && $downloadChoice <
1000000){ // A million is enough. I promise.
            $downloads = $downloadChoice;
        }

        // Download the object $downloads times from each bucket and time it to
demonstrate the speed difference.
        $directoryStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3ExpressService->getObject($directoryBucketName, $objectKey);
        }
        $directoryEndTime = hrtime(true);
        $directoryTimeDiff = $directoryEndTime - $directoryStartTime;

        $normalStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3RegularService->getObject($normalBucketName, $objectKey);
        }
        $normalEndTime = hrtime(true);
        $normalTimeDiff = $normalEndTime - $normalStartTime;

        echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
normal bucket took $normalTimeDiff.\n";
```

```
        echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
 " nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
 seconds.\n";
        pressEnter();

        // 7. Populate the buckets to show the lexicographical difference.
        echo "\n";
        echo "7. Populate the buckets to show the lexicographical difference.\n";
        echo "Now let's explore how Directory buckets store objects in a different
 manner to regular buckets.\n";
        echo "The key is in the name \"Directory!\"\n";
        echo "Where regular buckets store their key/value pairs in a flat manner,
 Directory buckets use actual directories/folders.\n";
        echo "This allows for more rapid indexing, traversing, and therefore
 retrieval times!\n";
        echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
        echo "This structural difference also causes ListObjects to behave
 differently, which can cause unexpected results.\n";
        echo "Let's add a few more objects with layered directories as see how the
 output of ListObjects changes.\n";
        pressEnter();

        // Populate a few more files in each bucket so that we can use ListObjects
 and show the difference.
        $otherObject = "other/$objectKey";
        $altObject = "alt/$objectKey";
        $otherAltObject = "other/alt/$objectKey";
        $s3ExpressService->putObject($directoryBucketName, $otherObject);
        $s3RegularService->putObject($normalBucketName, $otherObject);
        $this->resources['otherObject'] = $otherObject;
        $s3ExpressService->putObject($directoryBucketName, $altObject);
        $s3RegularService->putObject($normalBucketName, $altObject);
        $this->resources['altObject'] = $altObject;
        $s3ExpressService->putObject($directoryBucketName, $otherAltObject);
        $s3RegularService->putObject($normalBucketName, $otherAltObject);
        $this->resources['otherAltObject'] = $otherAltObject;

        $listDirectoryBucket = $s3ExpressService->listObjects($directoryBucketName);
        $listNormalBucket = $s3RegularService->listObjects($normalBucketName);

        // Directory bucket content
        echo "Directory bucket content\n";
        foreach($listDirectoryBucket['Contents'] as $result){
```

```
            echo $result['Key'] . "\n";
        }

        // Normal bucket content
        echo "\nNormal bucket content\n";
        foreach($listNormalBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        echo "Notice how the normal bucket lists objects in lexicographical order,
 while the directory bucket does not. This is because the normal bucket considers
 the whole \"key\" to be the object identifies, while the directory bucket actually
 creates directories and uses the object \"key\" as a path to the object.\n";
        pressEnter();

        echo "\n";
        echo "That's it for our tour of the basic operations for S3 Express One
 Zone.\n";
        $cleanUp = testable_readline("Would you like to delete all the resources
 created during this demo? Enter Y/y to delete all the resources.");
        if($cleanUp){
            $this->cleanUp();
        }



namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
```

```
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }

    public function setClient(S3Client $client)
    {
        $this->client = $client;
    }


    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
```

```
                echo "\nPlease fix error with bucket deletion before continuing.\n";
            }
            throw $exception;
        }
    }



    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
        }
    }



    public function putObject(string $bucketName, string $key, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $this->client->putObject($parameters);
            if ($this->verbose) {
                echo "Uploaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $key in $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with object uploading before continuing.";
            }
            throw $exception;
```

```
        }
    }



    public function getObject(string $bucketName, string $key, array $args = []):
 Result
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $object = $this->client->getObject($parameters);
            if ($this->verbose) {
                echo "Downloaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to download $key from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object downloading before continuing.";
            }
            throw $exception;
        }
        return $object;
    }



    public function copyObject($bucketName, $key, $copySource, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
 "CopySource" => $copySource], $args);
        try {
            $this->client->copyObject($parameters);
            if ($this->verbose) {
                echo "Copied the object from: $copySource in $bucketName to: $key.
\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to copy $copySource in $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object copying before continuing.";
            }
```

```
            throw $exception;
        }
    }



    public function listObjects(string $bucketName, $start = 0, $max = 1000, array
$args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
            throw $exception;
        }
        return $objects;
    }



    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
```

```
        return $contents;
    }



    public function deleteObjects(string $bucketName, array $objects, array $args =
[])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' => ['Objects'
=> $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
```

```
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteObject(string $bucketName, string $fileName, array $args =
 [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
```

```
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }



    public function preSignedUrl(CommandInterface $command, DateTimeInterface|int|
string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
```

```
    }

}
```

- API の詳細については、「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
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

SDK for PHP を使用した Amazon SES の例

次のコード例は、Amazon SES 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 の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おり、コードの設定方法と実行方法に関する手順を確認できます。

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。

SDK for PHP

を使用して AWS SDK for PHP 、Amazon RDS 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 E メールで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ます。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 RESTful PHP バックエンドと対話します。

- React.js ウェブアプリケーションを AWS サービスに統合します。
- Amazon RDS テーブル内の項目の一覧表示、追加、更新、削除を行います。
- Amazon SES を使用して、フィルター処理された作業項目の E メールレポートを送信します。
- 含まれている 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および実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

SDK for PHP を使用した Amazon SNS のコードサンプル

次のコード例は、Amazon SNS 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 の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おり、コードの設定方法と実行方法に関する手順を確認できます。

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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの CheckIfPhoneNumberIsOptedOut を参照してください。

ConfirmSubscription

次の例は、ConfirmSubscription を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ConfirmSubscription」を参照してください。

`CreateTopic`

次の例は、`CreateTopic` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「CreateTopic」を参照してください。

DeleteTopic

次の例は、DeleteTopic を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「DeleteTopic」を参照してください。

GetSMSAttributes

次の例は、GetSMSAttributes を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「GetSMSAttributes」を参照してください。

GetTopicAttributes

次の例は、GetTopicAttributes を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「GetTopicAttributes」を参照してください。

ListPhoneNumbersOptedOut

次の例は、ListPhoneNumbersOptedOut を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「ListPhoneNumbersOptedOut」を参照してください。

ListSubscriptions

次の例は、ListSubscriptions を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListSubscriptions」を参照してください。

ListTopics

次の例は、ListTopics を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「ListTopics」を参照してください。

Publish

次の例は、Publish を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「発行」を参照してください。

`SetSMSAttributes`

次の例は、`SetSMSAttributes` を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「SetSMSAttributes」を参照してください。

SetTopicAttributes

次の例は、SetTopicAttributes を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「SetTopicAttributes」を参照してください。

Subscribe

次の例は、Subscribe を使用する方法を説明しています。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 の詳細については、「AWS SDK for PHP API リファレンス」の「Subscribe」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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「Unsubscribe」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。

SDK for PHP

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設定および実行の手順については、GitHub で完全な例を参照してください。

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 Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

SMS テキストメッセージを発行する

次のコードサンプルは、Amazon SNS 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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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 の詳細については、AWS SDK for PHP API リファレンスの「発行」を参照してください。

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メッセージを取得し、各メッセージの内容を記録します。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用して Lambda で SNS イベントを消費します。

```
// 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 for PHP を使用した Amazon SQS の例

次のコード例は、Amazon SQS 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おり、コードの設定方法と実行方法に関する手順を確認できます。

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

次のコード例では、SQS 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け取る、Lambda 関数の実装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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での SQS イベントの消費。

```
// 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、SQS キュー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。

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

PHP を使用した Lambda での SQS バッチアイテム失敗のレポート。

```
// 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);
```

のバージョン 2 からの移行 AWS SDK for PHP

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

バージョン 3 の AWS SDK for PHP は、SDK の機能の向上、2 年以上にわたる顧客からのフィードバックの組み込み、依存関係のアップグレード、パフォーマンスの向上、最新の PHP 標準の導入に多大な労力を注いでいます。

バージョン 3 の新機能

のバージョン 3 は PSR-4 および PSR-7 標準 AWS SDK for PHP に従い、今後は SemVer 標準に従います。

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 との違いは何ですか？

プロジェクトの依存関係が更新される

このバージョンで SDK の依存関係が変更されています。

- SDK に PHP 8.1 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。これは、 AWS サービスにリクエストを送信するために SDK が使用する基盤となる HTTP クライアント実装を提供します。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 の Promises 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 の JMESPath 式」を参照してください。

リージョンとバージョンオプションが必要になりました

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。バージョン 2 では AWS SDK for PHP、 `'version'`は完全にオプションであり、場合によってはオプション`'region'`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確にすることで、コーディングする API バージョンと AWS リージョンをロックできます。新しい API バージョンが作成されるか、新しい AWS リージョンが利用可能になると、設定を明示的に更新する準備ができるまで、重大な変更の可能性から隔離されます。

> **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 サービスがすべての AWS リージョンで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含まれています。マルチリージョンクライアントは、コマンドパラメータ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する

のバージョン 3 では AWS SDK for PHP、クライアントのインスタンス化方法が変更されました。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ました

のバージョン 3 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 AWS SDK for PHP されました。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。その理由の 1 つは、開発者がプロジェクトに 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

のバージョン 3 では、 の代替として `Aws\Sdk` オブジェクト AWS SDK for PHP が導入されています`Aws\Common\Aws`。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 の結果が変更されました

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 DescribeEngineDefaultParameters を呼び出すと、バージョン 3 にラップする`EngineDefaults`」要素が含まれるようになりました。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)

列挙型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。

静的ファサードクラスが削除されました

バージョン 2 では AWS SDK for PHP、Laravel にヒントを得たあいまいな機能があり、 `Aws` クラス`enableFacades()`で を呼び出して、さまざまなサービスクライアントへの静的アクセスを有効にできました。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。

ページネーターはイテレーターよりも優先されます

のバージョン 2 には、* イテレーター* という機能 AWS SDK for PHP 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 = $s3Client->getIterator('ListObjects', ['Bucket' => 'amzn-s3-demo-bucket']);
foreach ($objects as $object) {
    echo $object['Key'] . "\n";
}
```

```
// Version 3
$results = $s3Client->getPaginator('ListObjects', ['Bucket' => 'amzn-s3-demo-bucket']);
foreach ($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 = $s3Client->getPaginator('ListObjects', ['Bucket' => 'amzn-s3-demo-bucket']);
foreach ($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。

多くの上位レベルの抽象化が変更されました

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 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のオプションと動作が少し変更されました。

  - `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。入門ガイドは、AWS PHP 開発ブログで確認できます。
  - Amazon S3 の `AcpBuilder` とその関連オブジェクトが削除されました。

SDK の両バージョンからのコードサンプルの比較

次の例は、 のバージョン 3 AWS SDK for PHP を使用する方法がバージョン 2 と異なる方法の一部を示しています。

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' => 'amzn-s3-demo-bucket',
```

```
        'Key'    => 'my-object-key'
    ]);

    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

SDK のバージョン 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' => 'amzn-s3-demo-bucket',
        'Key'    => 'my-object-key'
    ]);

    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
```

```
}
```

例: グローバル設定でクライアントをインスタンス化する

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 のバージョン 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.
```

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 お客様とお客様の間の責任共有です。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。における当社のセキュリティ責任は最優先事項であり 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ます。

クラウド内のセキュリティ – お客様の責任は、使用している AWS サービス、データの機密性、組織の要件、適用される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 バージョン 3 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
- AWS SDK for PHP バージョン 3 での Amazon S3 暗号化クライアントの移行

AWS SDK for PHP バージョン 3 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。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された AWS 責任共有モデルおよび GDPR の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 (IAM) 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。

この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- で 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。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- 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 のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」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を、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含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して AWS SDK for PHP バージョン 3 AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL 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 のサービス 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 リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理

- IAM AWS のサービス の操作方法
- 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 のサービス 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提供されます。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すると、管理者に適切なアクセス許可をリクエストするのに役に立ちます。の機能にアクセスできない場合は AWS、 AWS のサービス AWS ID とアクセスのトラブルシューティング「」または使用している 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。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で IAM を使用する方法の詳細については AWS、使用している AWS のサービス のユーザーガイドを参照してください。

IAM 管理者 - 管理者は、 AWSへのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM 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、 AWS のサービス 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 認証情報は、フェデレーション 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場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション 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、「 AWS サインイン ユーザーガイド」の「へのサインイン AWS アカウント方法」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を暗号化して署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに自分で署名する推奨方法の使用については、「IAM ユーザーガイド」の「API リクエストに対するAWS Signature Version 4」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。たとえば、 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 AWS 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「IAM ユーザーガイド」の「IAM のAWS 多要素認証」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 およびリソースへの AWS のサービス 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「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、ID プロバイダーとのフェデレーションを使用して一時的な認証情報 AWS のサービス を使用して にアクセスすることを要求します。

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、または ID ソースを介して提供された認証情報 AWS のサービス 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アイデンティティがアクセスすると AWS アカウント、ロールを引き受け、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か、独自の ID ソースのユーザーとグループのセッ

トに接続して同期し、すべての AWS アカウント とアプリケーションで使用できます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「What is IAM Identity Center?」(IAM Identity Center とは) を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、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 ID です。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」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 ID です。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。で IAM ロールを一時的に引き受けるには AWS Management Console、ユーザーから IAM ロール (コンソール) に切り替えることができます。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 URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド 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

については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center User Guide」の「Permission sets」を参照してください。

- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、 (プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」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 のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、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 のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする 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 のサービス 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アクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除することができます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」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 のサービス。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、サービスリンク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 EC2 で実行されているアプリケーション – 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 インスタンス内でのアクセスキーの保存に推奨されます。EC2 インスタンスに AWS ロールを割り当て、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、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」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、ID AWS またはリソースにアタッチします。ポリシーは AWS 、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) 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、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに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。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。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 API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 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

ティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」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。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

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では、IAM の AWS マネージド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 (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、および Amazon VPC は AWS WAF、ACLs。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的でない追加のポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「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) の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、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 AWS アカウント です。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、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を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー (SCP)」を参照してください。
- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM ポリシーを更新することなく、アカウント内のリソースに利用可能な最大数のアクセス許可を設定するために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースのアクセス許可を制限し、組織に属しているかどうかにかかわらず AWS アカウントのルートユーザー、 を含む ID の有効なアクセス許可に影響を与える可能性があります。RCP をサポートする のリストを含む Organizations と RCP の詳細については、AWS Organizations RCPs「リソースコントロールポリシー (RCPs」を参照してください。 AWS のサービス
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。が複数のポリシータイプが関与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、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

IAM AWS のサービス の操作方法

ほとんどの IAM 機能と AWS のサービス の連携方法の概要については、IAM ユーザーガイドのAWS「IAM と連携する のサービス」を参照してください。

IAM AWS のサービス で特定の を使用する方法については、関連する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 IAM の使用時に発生する可能性がある一般的な問題の診断 AWS と修正に役立ちます。

トピック

- でアクションを実行する権限がありません AWS
- iam:PassRole を実行する権限がありません
- 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ありません AWS

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、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、`mateojackson` IAM ユーザーがコンソールを使用して、ある *`my-example-widget`* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要な`awes:`*`GetWidget`* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 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:PassRole を実行する権限がありません

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して 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 のサービス 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、既存のロールをそのサービスに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。

以下の例のエラーは、`marymajor` という IAM ユーザーがコンソールを使用して AWSでアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。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。サインイン資格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 (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- 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[IAM AWS のサービス の操作方法](#)。
- 所有 AWS アカウント 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、「IAM 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」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介してアクセスを提供する方法については、「IAM ユーザーガイド」の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」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 のサービス 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、AWS のサービス 「コンプライアンスプログラムによる対象範囲内」を参照して、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 「 Compliance ProgramsAssurance」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、「Downloading Reports in AWS Artifact」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 のサービス 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 を提供します。

- セキュリティのコンプライアンスとガバナンス – これらのソリューション実装ガイドでは、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の機能をデプロイする手順を示します。
- HIPAA 対応サービスのリファレンス – HIPAA 対応サービスの一覧が提供されています。すべての AWS のサービス が HIPAA の対象となるわけではありません。
- 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。このガイドは、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) にわたるセキュリティコントロールを保護し、そのガイダンスに AWS のサービス マッピングするためのベストプラクティスをまとめたものです。
- 「 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、リソース設定が内部プラクティス、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 のサービス 、 内のセキュリティ状態を包括的に把握できます 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一覧については、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 – 不審な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モニタリングすることで AWS アカウント、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。GuardDuty を使用すると、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、PCI DSS などのさまざまなコンプライアンス要件に対応できます。
- AWS Audit Manager – これにより AWS のサービス 、 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 、コンプライアンスプログラムによる AWS 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、複数の物理的に分離および分離されたアベイラビリティーゾーンを提供します。これらは、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 、コンプライアンスプログラムによる AWS 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 AWS 保護されています。 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。

AWS 公開された API コールを使用して、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)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードはJava 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。またはAWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

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 、コンプライアンスプログラムによる AWS 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。

AWS SDK for PHP バージョン 3 での 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 AWS SDKs、各 SDK を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として、V1 復号化クライアントを最新の SDK リリースに更新します。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については、Composer ファイルで SDK パッケージを SDK のバージョン 3.148.0 に更新し、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求められる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、キーと値のペアの連想配列であり、 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 `true` に設定すると、KMS キーを指定することなく復号化が可能になります。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: kms-key-id AWS KMS で を使用する

> **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。

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 は、サービスの説明と動的マジック __call() メソッドを使用して API オペレーションを実行します。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、 AWS SDK for PHP は PHP のコンパイル時に設定された CA バンドルを使用します。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 AWS 全般のリファレンス。

```
// 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。

これを行う方法の完全な例については、AWS SDK for PHP 「バージョン 3 でのカスタム Amazon CloudSearch ドメインリクエストの署名」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?

の使用`Aws\Exception\CredentialsException`中に が表示される場合は AWS SDK for PHP、SDK に認証情報が提供されておらず、環境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。

は HHVM で AWS SDK for PHP 動作しますか？

は現在 HHVM で実行 AWS SDK for PHP されず、HHVM のイールドセマンティクスの問題が解決されるまで を実行できません。

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

```
$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、システムをチェックし、AWS SDK for PHP バージョン 3 の SDK 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。次に、 AWS SDK for PHP はこれをx-amz-content-sha-256」ヘッダーとして使用し、正規リクエストの本文チェックサムとして使用します。

```
$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 また、 は、再試行スキームのバックオフおよびジッターアルゴリズムと指数遅延を統合します。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カスタマイズされた`Exception`クラスに加えて、各 AWS サービスクライアントには、AwsExceptionAwsException から継承する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。クライアントの使用を開始するthe section called “基本的なサービスクライアントの作成”には、「」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。リクエストの作成セクションで説明されているようにサービスオペレーションを実行する場合、コマンドオブジェクトは直接処理されません。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 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。

Operation

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はページ分割され、切り捨てられた結果で応答します。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 ガイドのページネーター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 ライブラリ全体を参照できますが、各サービスのクライアントオブジェクトのファクトリとして機能する `Aws\Sdk` クラス (ドキュメント) も参照でき

ます。 https://docs.aws.amazon.com/aws-sdk-php/v3/api/class-Aws.Sdk.htmlSdk 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

サービス AWS (Amazon S3、Amazon DynamoDB、 AWS OpsWorks など) を参照する一般的な方法。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 ガイドのウェーターを参照してください。

最新の AWS 用語については、「」のAWS 「用語集」を参照してください AWS 全般のリファレンス。

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
目次の更新	他の SDK ガイドとの整合性を高めるために目次を改訂しました。	2025 年 7 月 28 日
認証情報トピックのリビジョン	トピックが再編成されました。デフォルトの認証情報プロバイダーチェーンの詳細が表示されます。	2025年1月10日
Amazon S3 マルチパートアップロード	`ObjectUploader` および のサブコマンドの設定に使用できる「params」配列を文書化します。 `MultipartUploader`	2024 年 11 月 6 日
オブジェクトアップローダー	S3 アップロード`ObjectUploader` の で利用可能なコールバックの使用を明確にする	2024 年 10 月 11 日
Amazon S3 バケット名を更新する	ガイド全体で S3 バケット名を更新しました。	2024 年 9 月 30 日
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 共通ランタイム (AWS CRT) の	2023 年 11 月 17 日

	使用について説明するトピックを追加します。	
StreamWrapper mkdir() の最新情報	`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
基本的な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 SDKs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に複数のリンクを追加しました。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 SDKsおよびツールリファレンスガイドへのリファレンスを追加しました。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	GitHub で利用できるコード例のリストを含めました。	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

変更	説明	リリース日
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 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 の例	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	AWS スタイルに切り替えたドキュメント	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 のサポートの追加	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 Service と Amazon CloudSearch	サービスの例の追加	2015 年 12 月 28 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 および のサービス機能を追加 AWS	2015 年 4 月 30 日

変更	説明	リリース日
SDK の新バージョン	のバージョン 3 が AWS SDK for PHP リリースされました。	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。